RUTGERS
THE STATE UNIVERSITY OF NEW JERSEY

MITA CAPSTONE PROJECT
"Instacart Market Basket Analysis"

Under the guidance of Professor Dr. Hui Xiong

Project report submitted by

Valencia Dias-196004354

TABLE OF CONTENTS

1.INTRODUCTION:

Instacart is an e-commerce website that allows users to shop for groceries from a local grocery store online, and then sends an Instacart personal shopper to pick up and deliver the orders made by users the same day. Instacart uses transactional data to develop models that predict which products a user will buy again, try for the first time, or add to their cart next during a session. These processes allow retailers to conduct analysis on purchase iterations by users but understanding the customer purchasing patterns and behaviors can become tedious and challenging. There are three ways that Instacart generates revenue: delivery fees, membership fees, and mark-ups on in-store prices.

1.1 Goals and Objectives:

The main objective of the project is to predict what the user will buy in the next order, given all data of prior orders.

2. Related Work:

Recommendation systems have been widely and fruitfully studied in recent years. The 2009 Netflix Prize collaborative filtering competition is the most famous of many studies examining the best way to recommend products and services to consumers. Grocery recommendations are a tougher nut to crack. Unlike Netflix, which has a limited number of movies and TV shows to credibly recommend, grocery shopping presents a challenge in its high sparsity. A grocery store stocks thousands of items, yet most people only buy a handful of them at a time. Analysis of this question has included methods like basket-sensitive random walks and SVD approximations to recommend items to consumers. Others have delved more into the theory, hoping to lay out a process that incorporates both product and user features into a recommendation process.

3.Dataset Preparation:

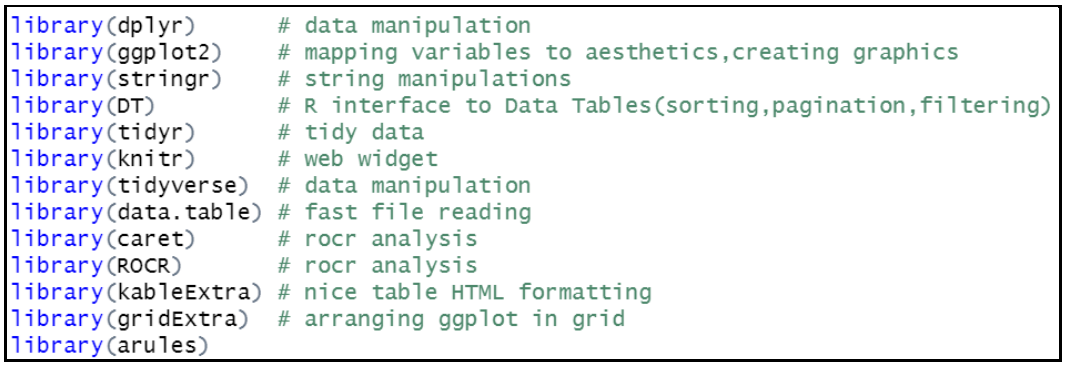
3.1 Data Source

“The Instacart Online Grocery Shopping Dataset 2017” was released by Instacart as a public dataset. The dataset contains over 3 million anonymized grocery orders from more than 200,000 Instacart users. The following analysis will make use of this datasets.

Data source link: https://www.kaggle.com/c/instacart-market-basket-analysis/data

3.2 R Libraries Used

Here are the R libraries used in the analysis:



```
library(dplyr)      # data manipulation
library(ggplot2)    # mapping variables to aesthetics,creating graphics
library(stringr)    # string manipulations
library(DT)         # R interface to Data Tables(sorting,pagination,filtering)
library(tidyr)      # tidy data
library(knitr)      # web widget
library(tidyverse)  # data manipulation
library(data.table) # fast file reading
library(caret)      # rocr analysis
library(ROCR)       # rocr analysis
library(kableExtra) # nice table HTML formatting
library(gridExtra)  # arranging ggplot in grid
library(arules)
```

3.3 Data Dictionary:

The dataset is a relational set of files describing customers' orders over time. They are anonymized and contains a sample of over **3 million grocery orders** from more than **200,000 Instacart users**. For each user, Instacart provided **between 4 and 100** of their orders, with the sequence of products purchased in each order, the **week and hour of day** the order was placed, and a **relative measure of time between orders**.

Total six datasets were imported. Following section will explore each dataset in further detail. These datasets were sourced from an existing Kaggle competition.
orders (3.4m rows, 206k users):

- order_id: order identifier
- user_id: customer identifier
- eval_set: which evaluation set this order belongs in (see SET described below)
- order_number: the order sequence number for this user (1 = first, n = nth)
- order_dow: the day of the week the order was placed on
- order_hour_of_day: the hour of the day the order was placed on
- days_since_prior: days since the last order, capped at 30 (with NAs for order_number = 1)

products (50k rows):

- product_id: product identifier
- product_name: name of the product
- aisle_id: foreign key
- department_id: foreign key

aisles (134 rows):

- aisle_id: aisle identifier
- aisle: the name of the aisle

departments (21 rows):

- department_id: department identifier
- department: the name of the department

order_products__SET (30m+ rows):

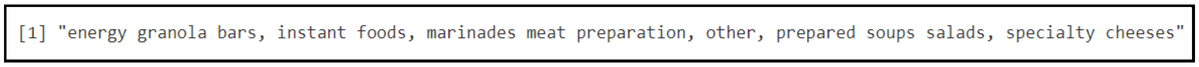
- order_id: foreign key
- product_id: foreign key
- add_to_cart_order: order in which each product was added to cart

- reordered: 1 if this product has been ordered by this user in the past, 0 otherwise

where SET is one of the four following evaluation sets (eval_set in orders):
- "prior": orders prior to that users most recent order (~3.2m orders)
- "train": training data supplied to participants (~131k orders)
- "test": test data reserved for machine learning competitions (~75k orders)

3.4 Understanding Data:
3.5.1 Aisles:

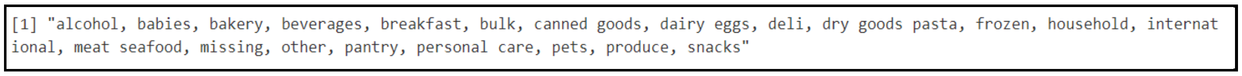
There are **134 aisles** in this dataset. Here are few sample names of the aisles.



```
[1] "energy granola bars, instant foods, marinades meat preparation, other, prepared soups salads, specialty cheeses"
```

3.5.2 Departments:

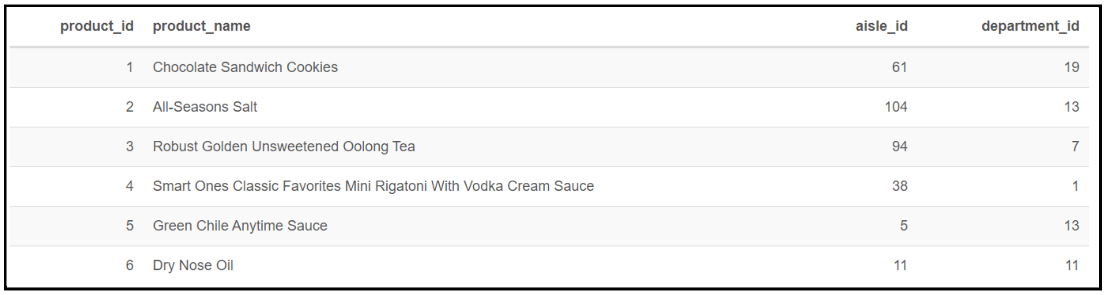
There are **21 departments** in this dataset. Names of all departments are listed below in alphabetically ordered.



```
[1] "alcohol, babies, bakery, beverages, breakfast, bulk, canned goods, dairy eggs, deli, dry goods pasta, frozen, household, internat
ional, meat seafood, missing, other, pantry, personal care, pets, produce, snacks"
```

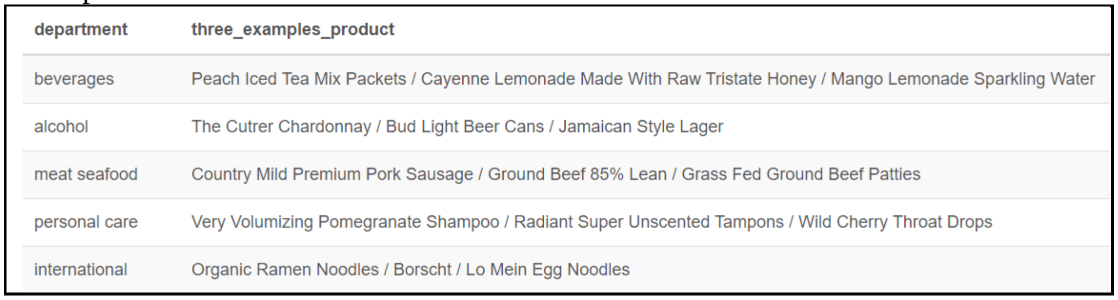
3.5.3 Products:

There are **49,688 products** in the catalogue within **134 aisles and 21 departments**. Sample products are as below



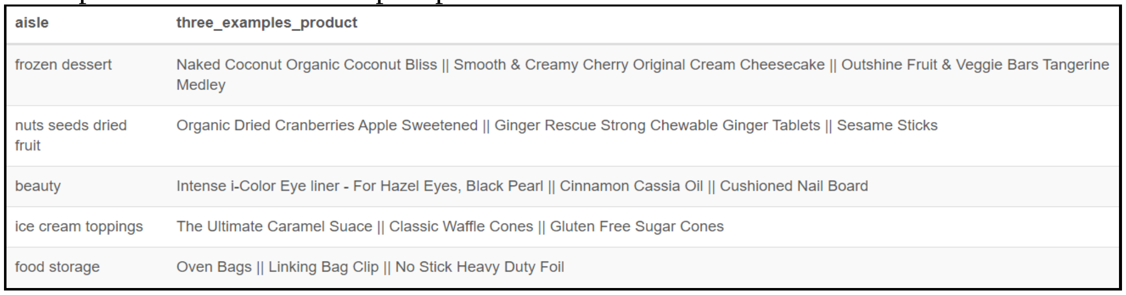
product_id	product_name	aisle_id	department_id
1	Chocolate Sandwich Cookies	61	19
2	All-Seasons Salt	104	13
3	Robust Golden Unsweetened Oolong Tea	94	7
4	Smart Ones Classic Favorites Mini Rigatoni With Vodka Cream Sauce	38	1
5	Green Chile Anytime Sauce	5	13
6	Dry Nose Oil	11	11

- **Departments and its Relevant Products:**
  Products dataframe is related to Departments. We shall see sample of 3 products for few departments



department	three_examples_product
beverages	Peach Iced Tea Mix Packets / Cayenne Lemonade Made With Raw Tristate Honey / Mango Lemonade Sparkling Water
alcohol	The Cutrer Chardonnay / Bud Light Beer Cans / Jamaican Style Lager
meat seafood	Country Mild Premium Pork Sausage / Ground Beef 85% Lean / Grass Fed Ground Beef Patties
personal care	Very Volumizing Pomegranate Shampoo / Radiant Super Unscented Tampons / Wild Cherry Throat Drops
international	Organic Ramen Noodles / Borscht / Lo Mein Egg Noodles

- **Aisles and Its Relevant Products:**
  Products dataframe is also related to aisles. Each aisle relates to multiple products. By joining both aisles and products dataframe, we have an idea what type of products for each aisle.
  Example below shows 3 samples products of for few aisles.



aisle	three_examples_product
frozen dessert	Naked Coconut Organic Coconut Bliss || Smooth & Creamy Cherry Original Cream Cheesecake || Outshine Fruit & Veggie Bars Tangerine Medley
nuts seeds dried fruit	Organic Dried Cranberries Apple Sweetened || Ginger Rescue Strong Chewable Ginger Tablets || Sesame Sticks
beauty	Intense i-Color Eye liner - For Hazel Eyes, Black Pearl || Cinnamon Cassia Oil || Cushioned Nail Board
ice cream toppings	The Ultimate Caramel Suace || Classic Waffle Cones || Gluten Free Sugar Cones
food storage	Oven Bags || Linking Bag Clip || No Stick Heavy Duty Foil

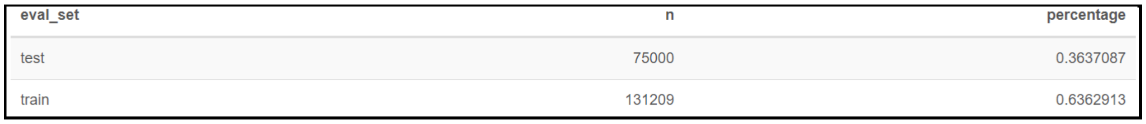
3.5.4 Orders:

There are over **3 million** observations in orders dataset. Each row represents an **unique order**.

- **Train Eval_Set**
  Let's analyze the construct of one user. For example, **user_id 1** had made **10 prior orders** (order number from 1 to 10), last order is a **train** (eval_set). Note that the first order (order_number 1) does not have value for day_since_prior_order, as it is the first order without prior records.
  This also means **<user_id, product_id>** made up the **key** for prediction.
- **Test Eval_Set**
  Let's analyze another construct of orders. **User_id 3** had made **12 orders** before the final order labeled as **test** (eval_set) order. From the data we know that order_number is being recycled for each user.
  Instacart did not provide us the basket content for test order. This is in fact the **target for prediction**.
- **Order_Product**
  Each order contains multiple products purchased by user. Instacart had cleanly categorized the orders into 'train' and 'prior' in **SINGLE** order dataset.
  However, the detail of each orders is splitted into two datasets:
  - **order_product_train**: contain only detail product items of last order
  - **order_product_prior**: contain detail product items of all prior orders

3.5.5 Users

There is no dedicated dataframe for users. However, we can derive number of unique users from **order** dataframe. By grouping the user_id and eval_set column, we found that there are **75,000 test** users, **131,209 train** users.



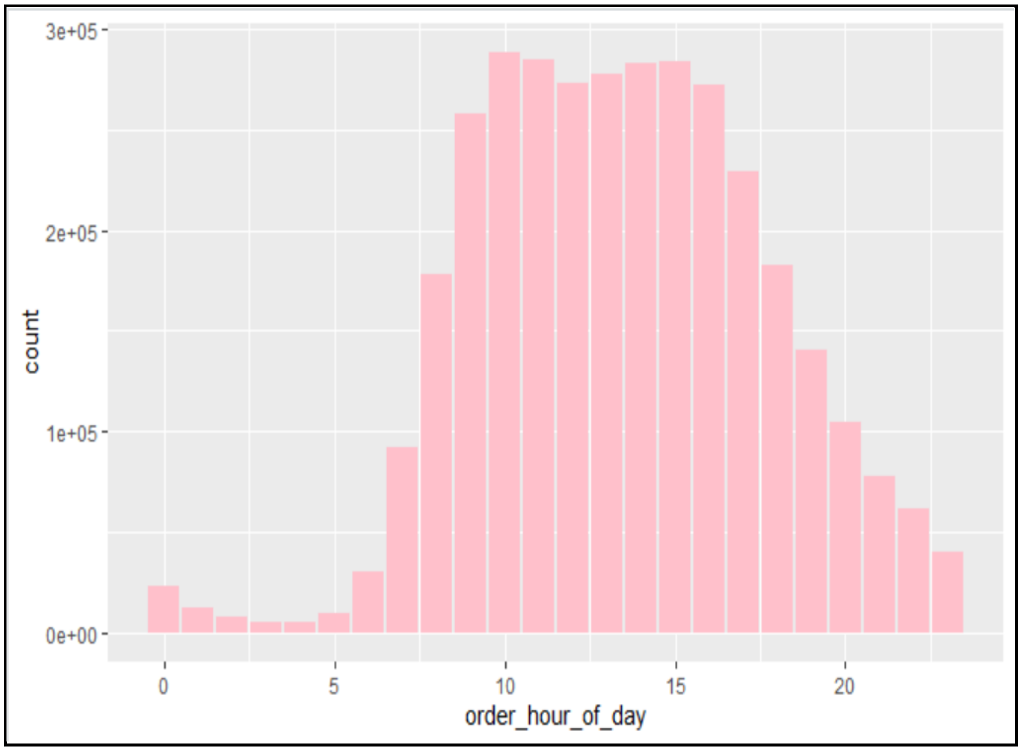
eval_set	n	percentage
test	75000	0.3637087
train	131209	0.6362913

4.Exploratory Data Analysis:

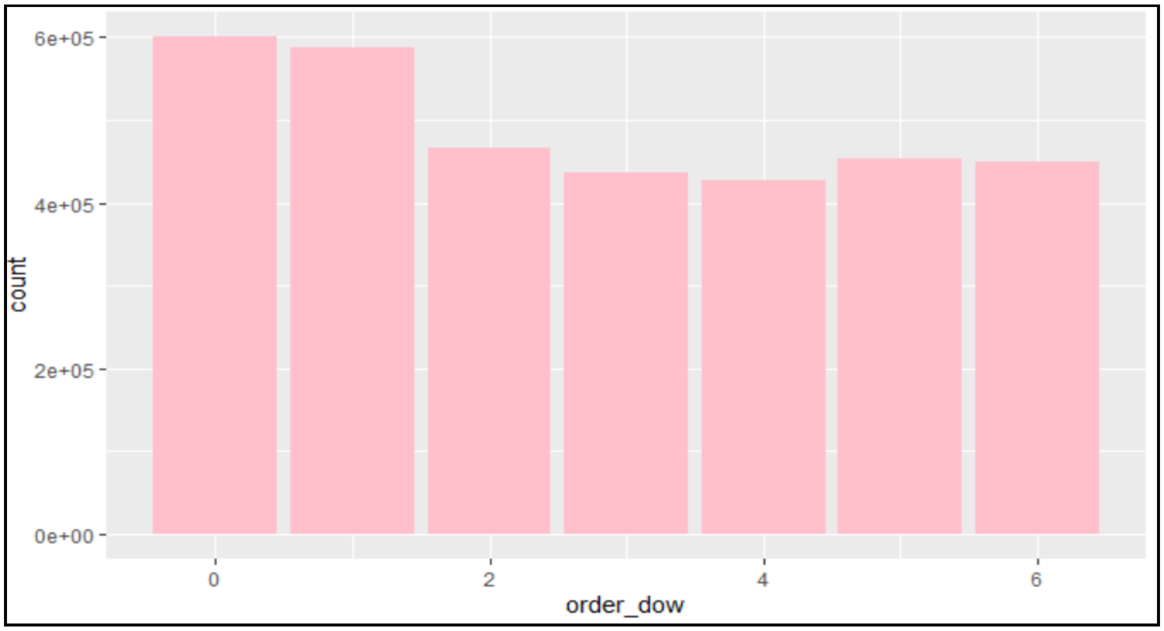
In this section, I analyzed the buying behavior by asking some interesting questions.

- When do they buy (day and time)?
- What usually do people buy?
- Which one they usually reorder?
- Is there a buying trend and does it influence what they buy?

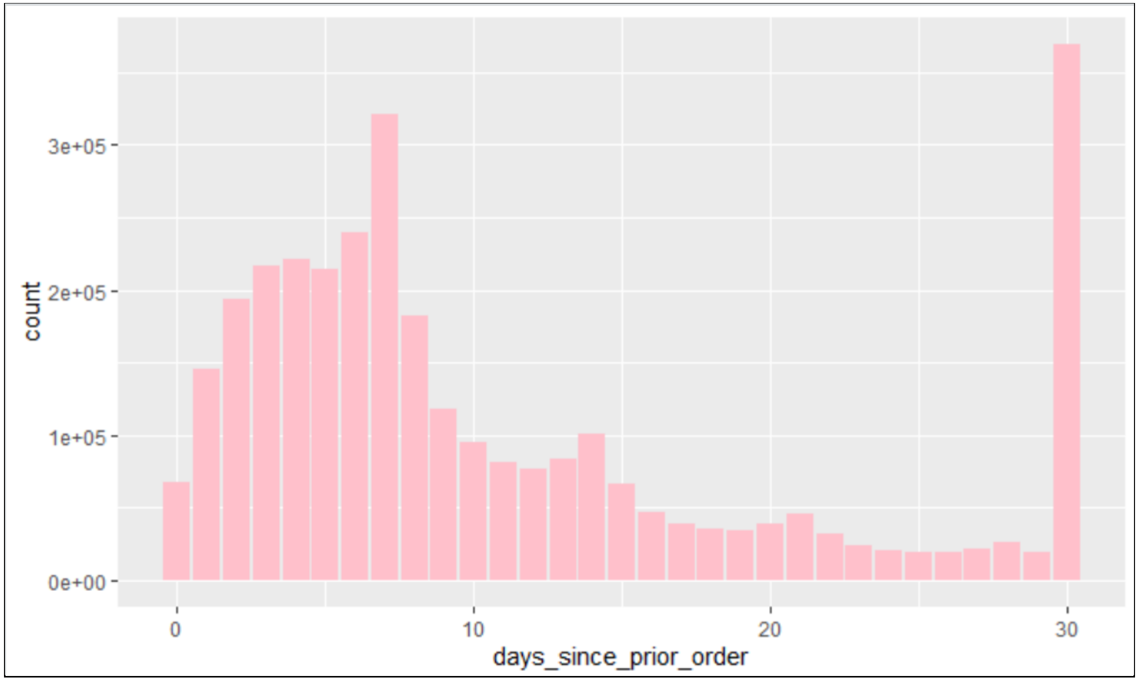
- **When do people buy?**
  -Hour of Day
  There is a clear effect of hour of day on order volume.
  Most orders are between 8.00-18.00



-**Day of Week**
There is a clear effect of day of the week. Most orders are on days 0 and 1. Unfortunately there is no info regarding which values represent which day, but one would assume that this is the weekend.

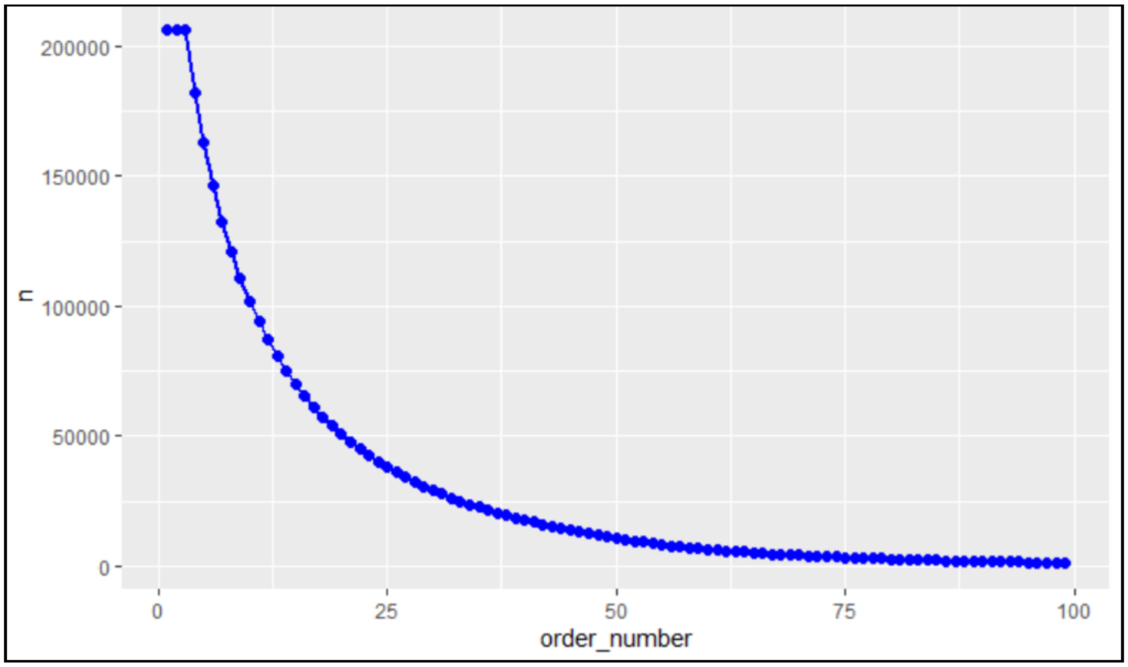


- **When do they order again?**
  People seem to order more often after exactly 1 week.



- **How many prior orders are there?**

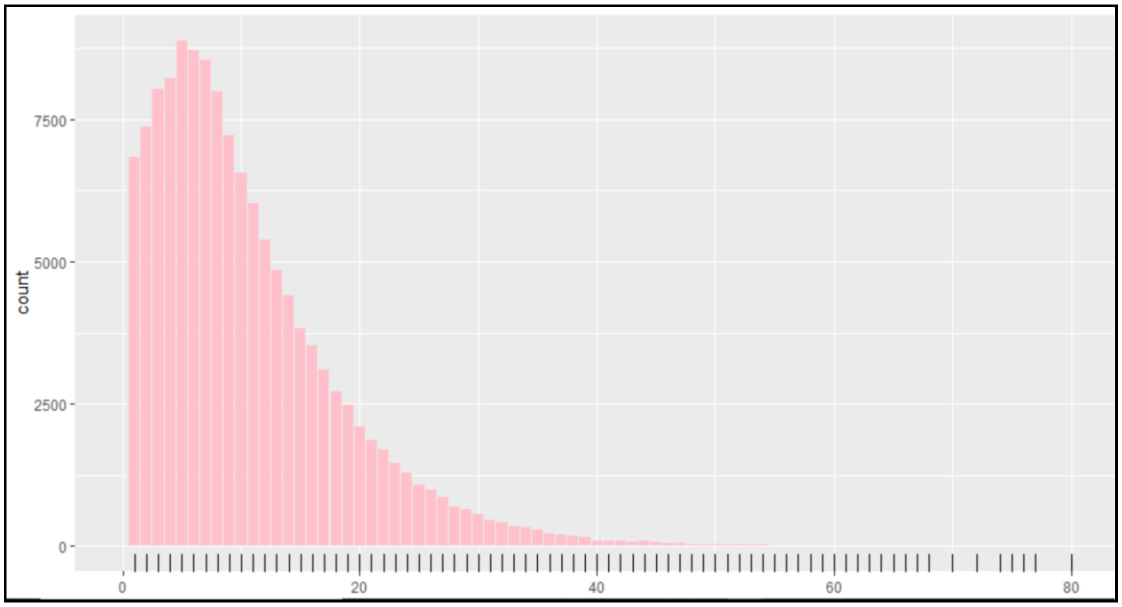
  We can see that there are always at least 3 prior orders.



- **How many items do people buy?**

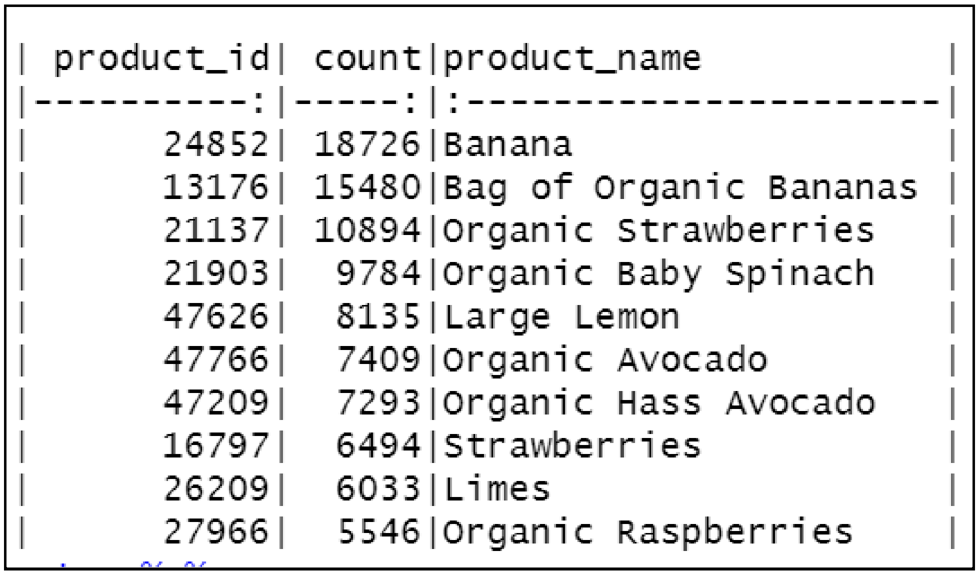
  Let us have a look how many items are in the orders. We can see that people most often order around 5 items.

  The distributions are comparable between the train and prior order set.

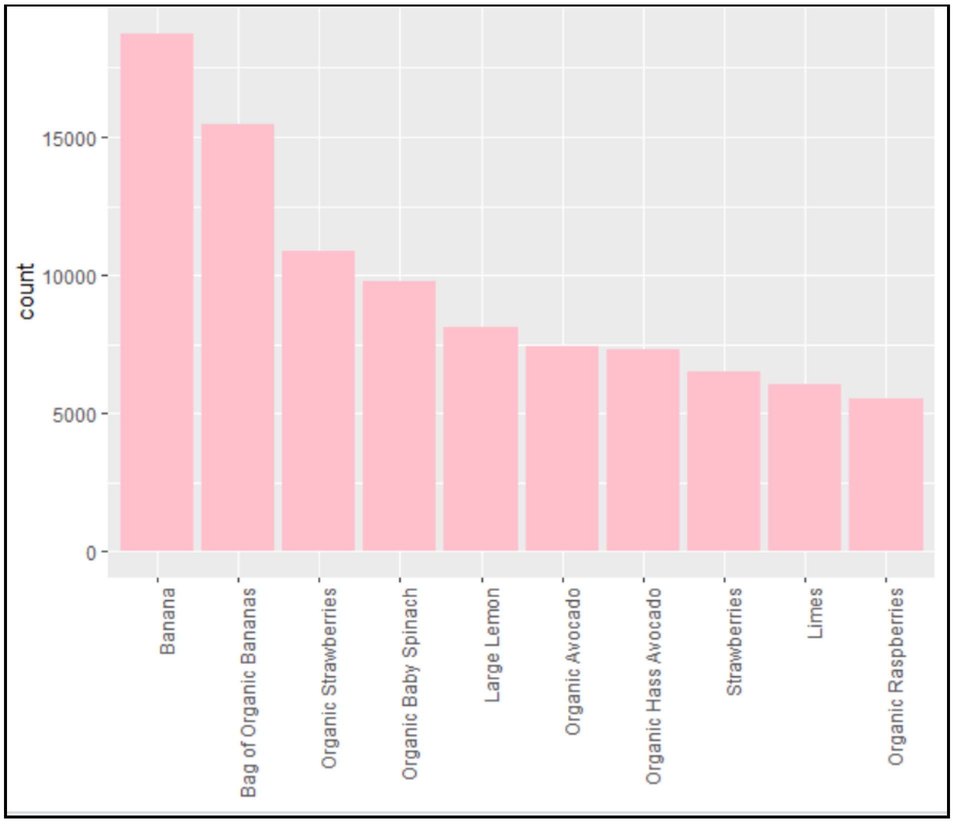


- **Bestsellers**

  Let us have a look at which products are sold most often (top10). And the clear winner is: **Bananas**

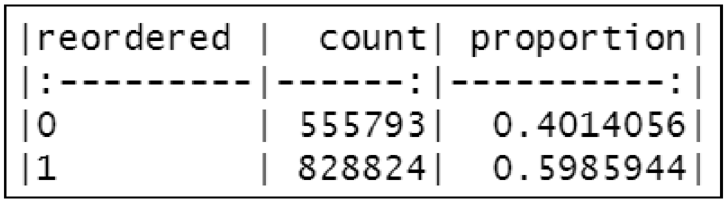


```
| product_id| count|product_name          |
|----------:|-----:|:---------------------|
|      24852| 18726|Banana                |
|      13176| 15480|Bag of Organic Bananas |
|      21137| 10894|Organic Strawberries  |
|      21903|  9784|Organic Baby Spinach  |
|      47626|  8135|Large Lemon           |
|      47766|  7409|Organic Avocado       |
|      47209|  7293|Organic Hass Avocado  |
|      16797|  6494|Strawberries          |
|      26209|  6033|Limes                 |
|      27966|  5546|Organic Raspberries   |
```

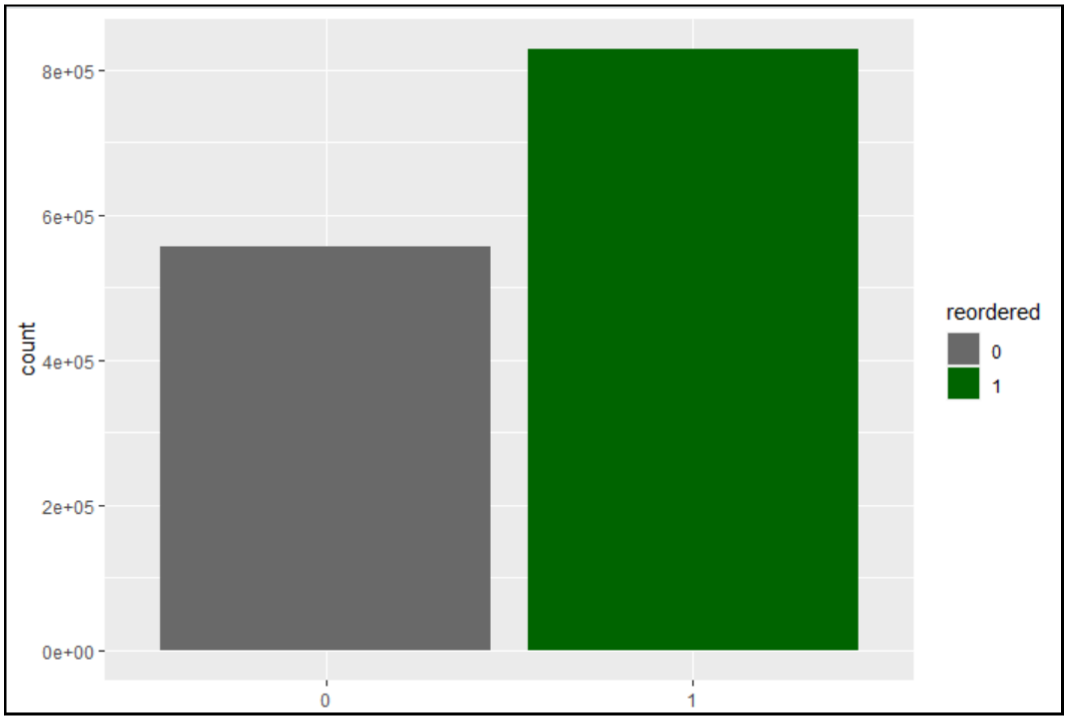


- **How often do people order the same items again?**

  59% of the ordered items are reorders.

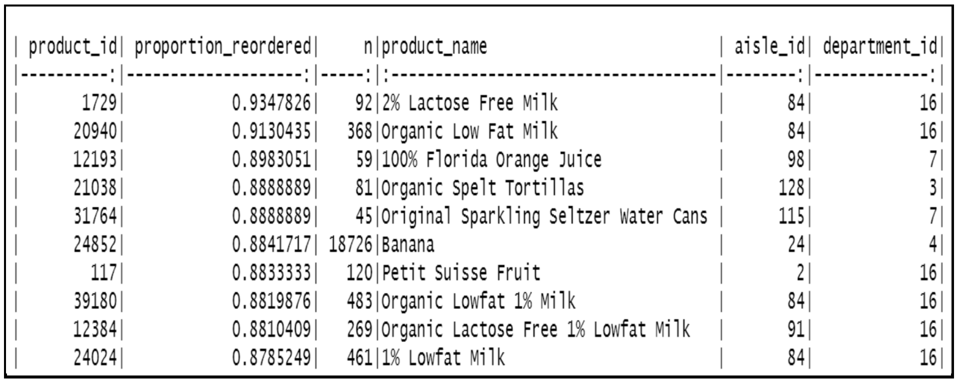


```
|reordered |  count| proportion|
|:---------|------:|----------:|
|0         | 555793|  0.4014056|
|1         | 828824|  0.5985944|
```

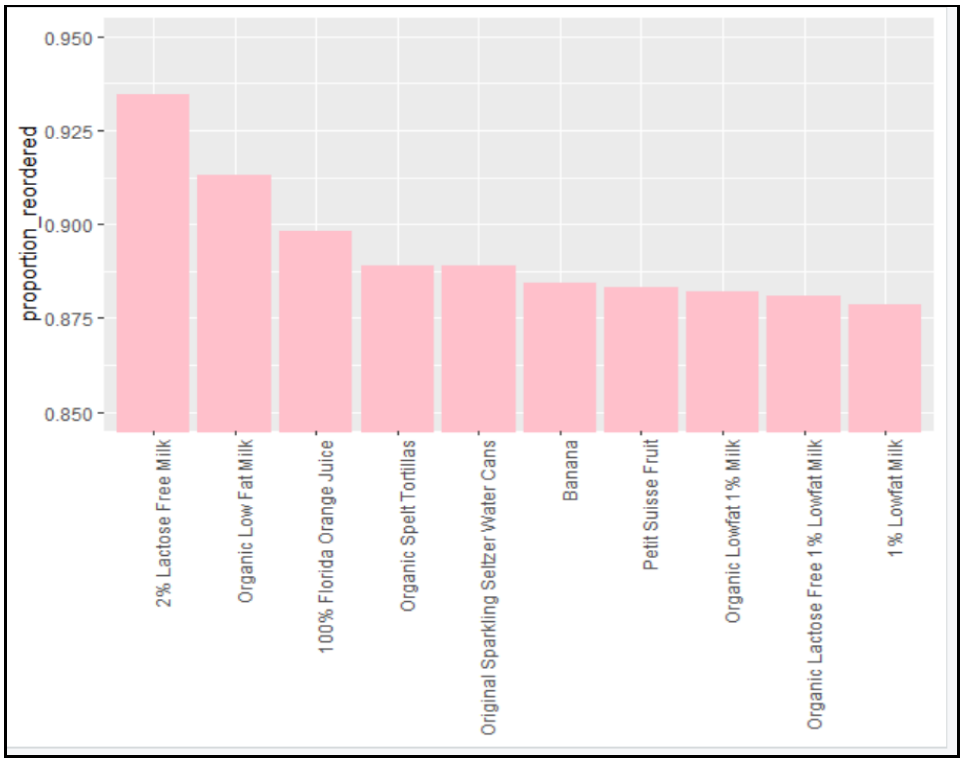


- **Most often reordered**

  Now here it becomes interesting. These 10 products have the highest probability of being reordered.

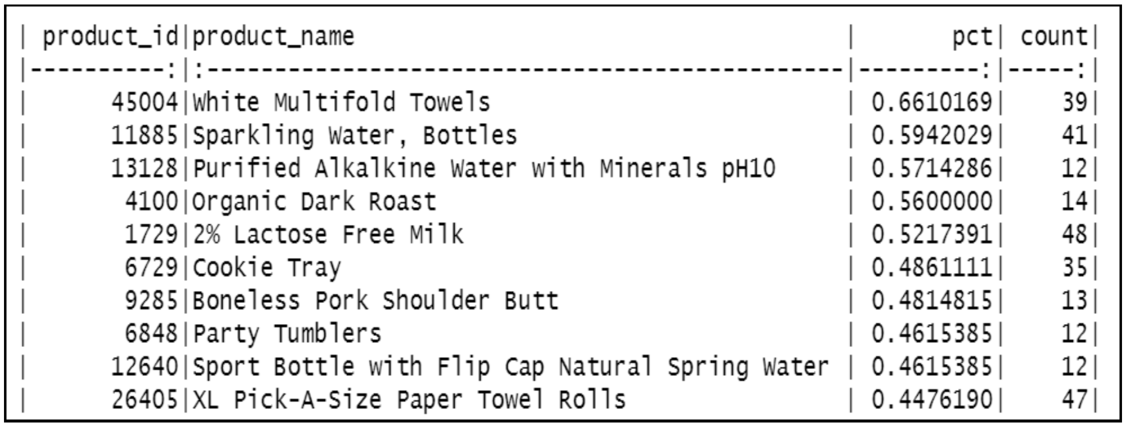


```
| product_id| proportion_reordered|     n|product_name                        | aisle_id| department_id|
|----------:|--------------------:|-----:|:-----------------------------------|--------:|-------------:|
|       1729|            0.9347826|    92|2% Lactose Free Milk                |       84|            16|
|      20940|            0.9130435|   368|Organic Low Fat Milk                |       84|            16|
|      12193|            0.8983051|    59|100% Florida Orange Juice           |       98|             7|
|      21038|            0.8888889|    81|Organic Spelt Tortillas             |      128|             3|
|      31764|            0.8888889|    45|Original Sparkling Seltzer Water Cans |      115|             7|
|      24852|            0.8841717| 18726|Banana                              |       24|             4|
|        117|            0.8833333|   120|Petit Suisse Fruit                  |        2|            16|
|      39180|            0.8819876|   483|Organic Lowfat 1% Milk              |       84|            16|
|      12384|            0.8810409|   269|Organic Lactose Free 1% Lowfat Milk |       91|            16|
|      24024|            0.8785249|   461|1% Lowfat Milk                      |       84|            16|
```

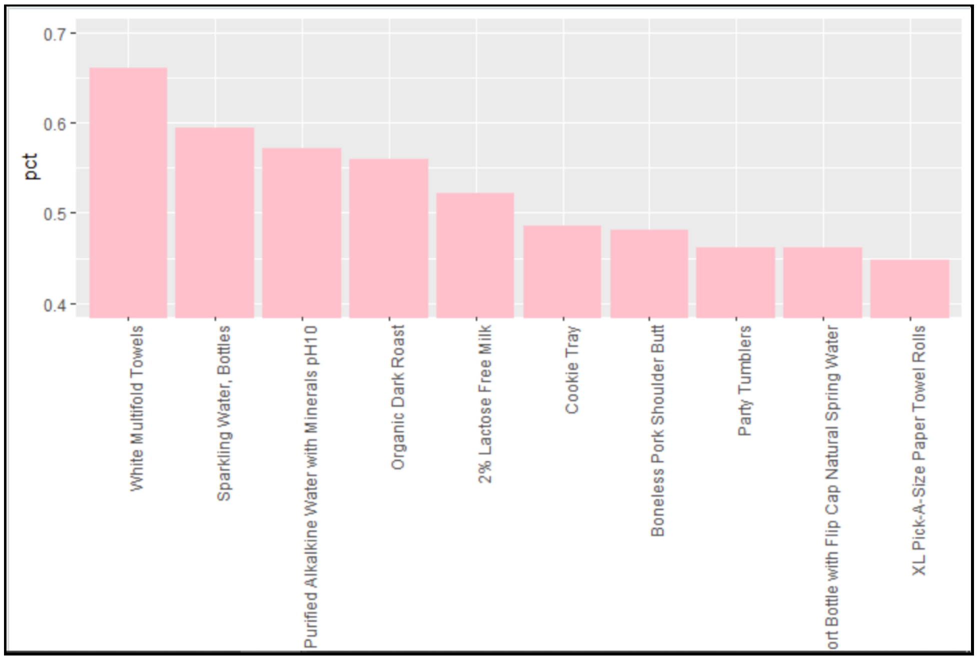


- **Which item do people put into the cart first?**

  People seem to be quite certain about Multifold Towels and if they buy them, put them into their cart first in 66% of the time.

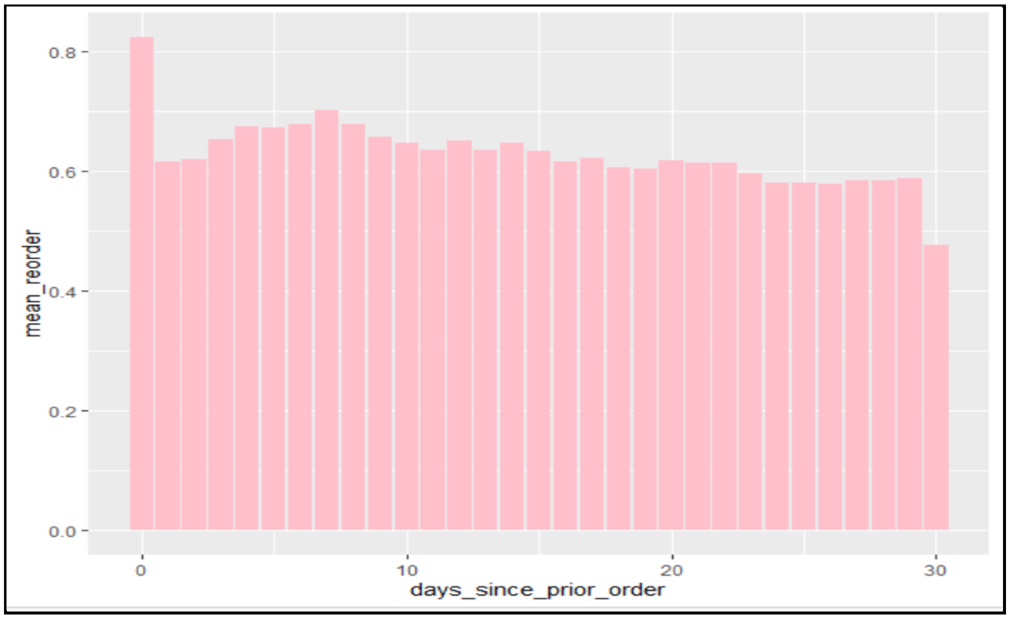


```
| product_id|product_name                                  |       pct| count|
|----------:|:---------------------------------------------|---------:|-----:|
|      45004|White Multifold Towels                        | 0.6610169|    39|
|      11885|Sparkling Water, Bottles                      | 0.5942029|    41|
|      13128|Purified Alkalkine Water with Minerals pH10   | 0.5714286|    12|
|       4100|Organic Dark Roast                            | 0.5600000|    14|
|       1729|2% Lactose Free Milk                          | 0.5217391|    48|
|       6729|Cookie Tray                                   | 0.4861111|    35|
|       9285|Boneless Pork Shoulder Butt                   | 0.4814815|    13|
|       6848|Party Tumblers                                | 0.4615385|    12|
|      12640|Sport Bottle with Flip Cap Natural Spring Water | 0.4615385|    12|
|      26405|XL Pick-A-Size Paper Towel Rolls              | 0.4476190|    47|
```



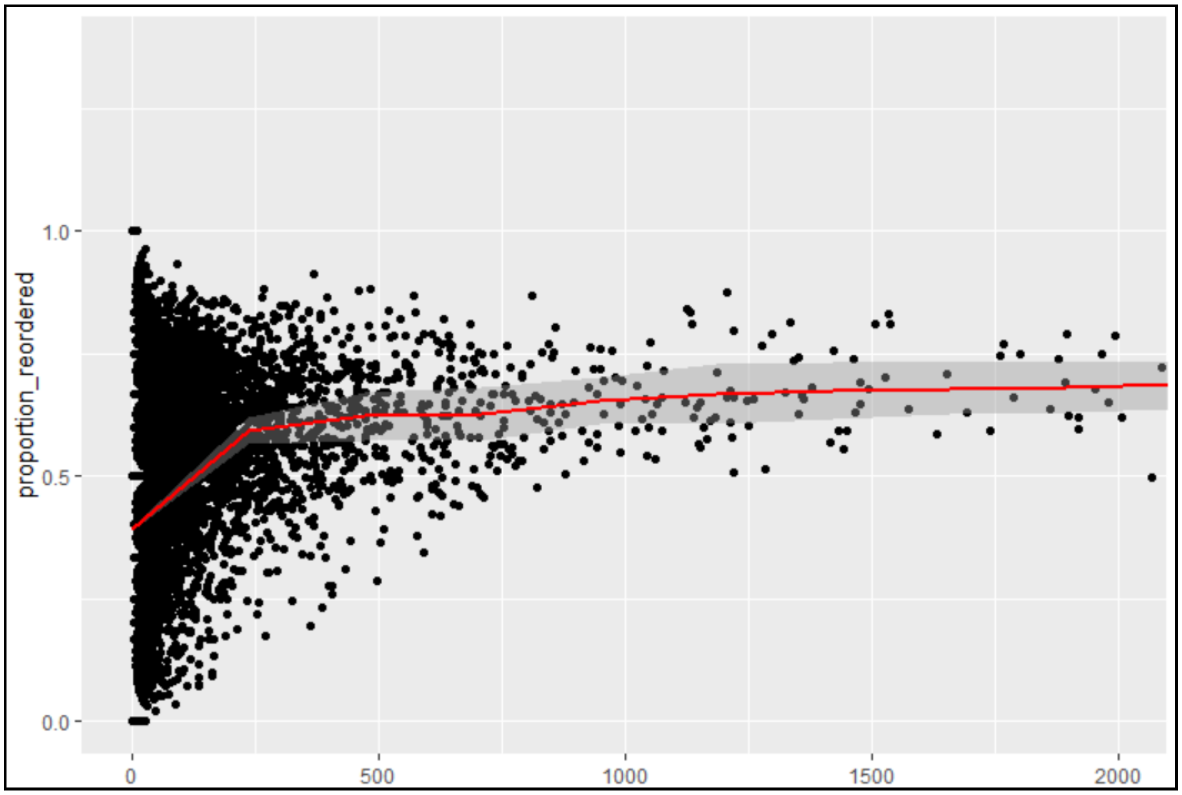
- **Association between time of last order and probability of reorder**

  This is interesting: We can see that if people order again on the same day, they order the same product more often. Whereas when 30 days have passed, they tend to try out new things in their order.



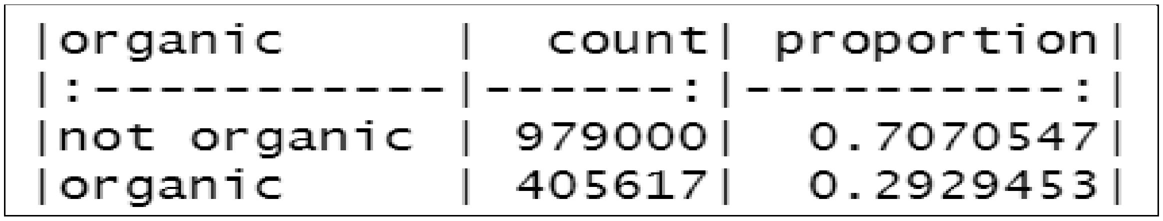
- **Association between number of orders and probability of reordering**

  Products with a high number of orders are naturally more likely to be reordered. However, there seems to be a ceiling effect.

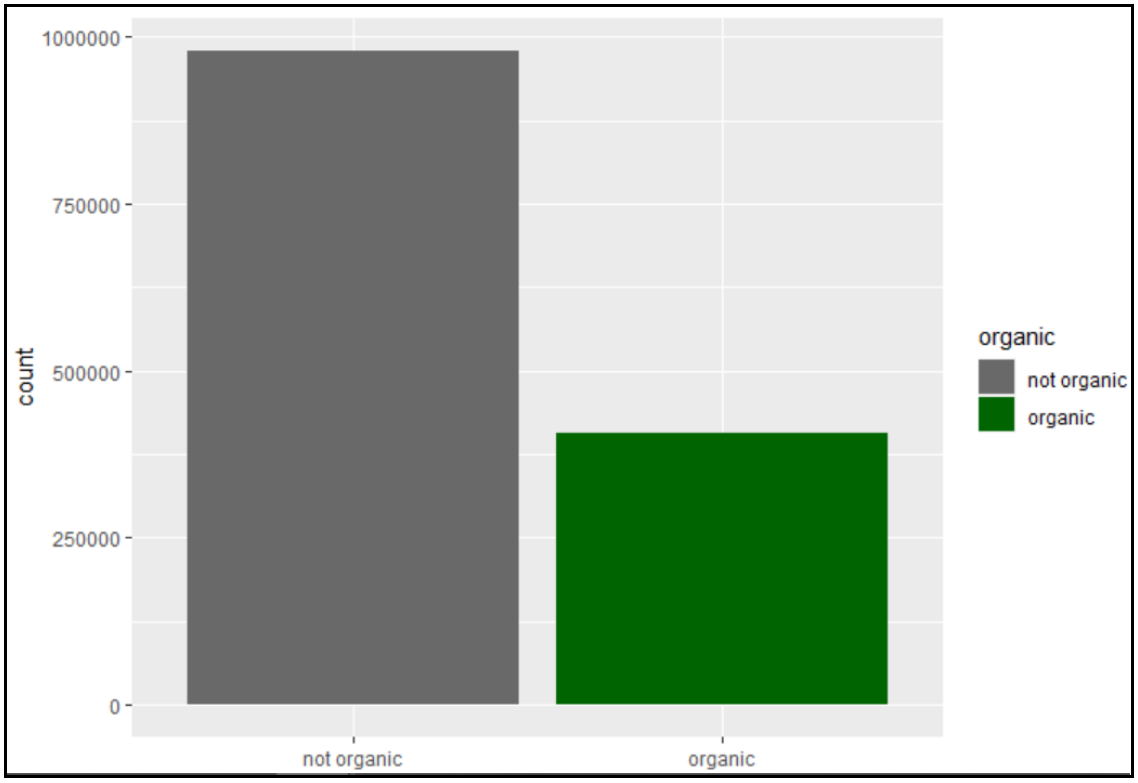


- **Organic Vs Non-Organic**

  What is the percentage of orders that are organic vs. not organic?

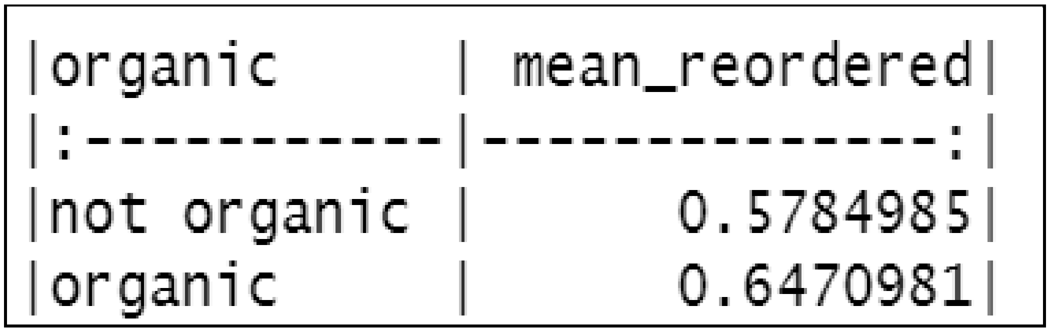


```
|organic      |   count| proportion|
|:------------|------:|----------:|
|not organic  | 979000|  0.7070547|
|organic      | 405617|  0.2929453|
```

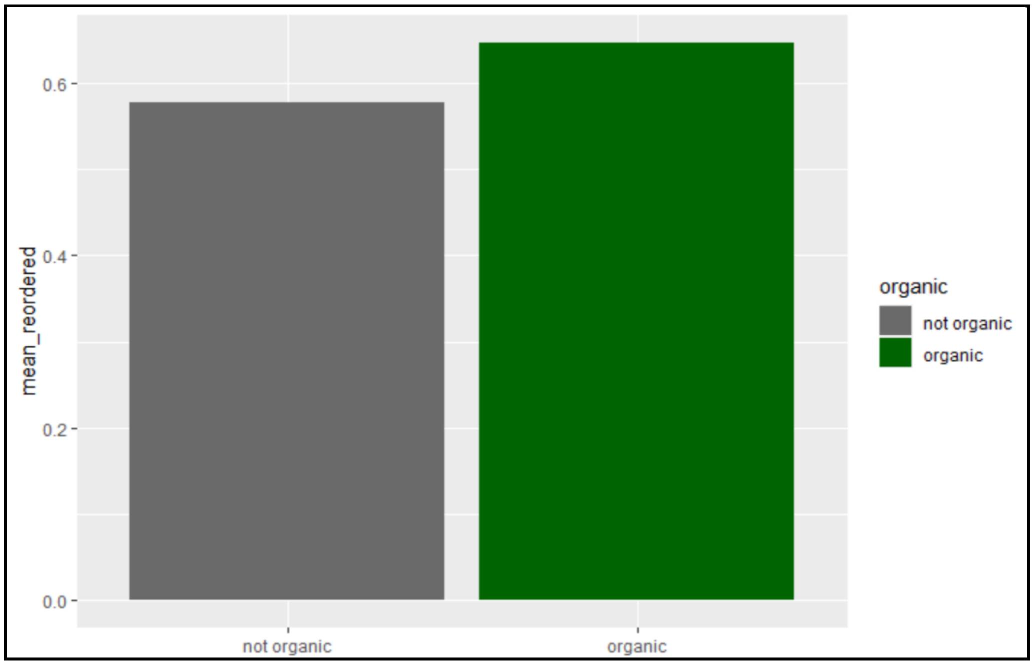


- **Reordering Organic vs Non-Organic**

  People more often reorder organic products vs non-organic products.



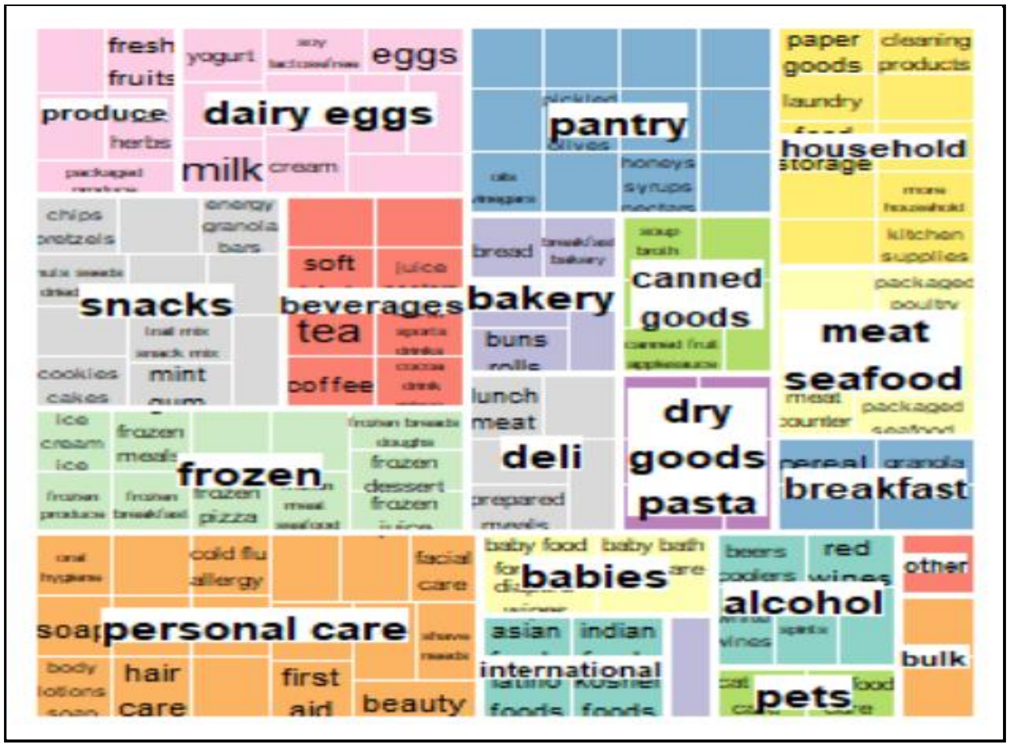
```
|organic      | mean_reordered|
|:------------|--------------:|
|not organic  |      0.5784985|
|organic      |      0.6470981|
```



- **Visualizing the Product Portfolio**

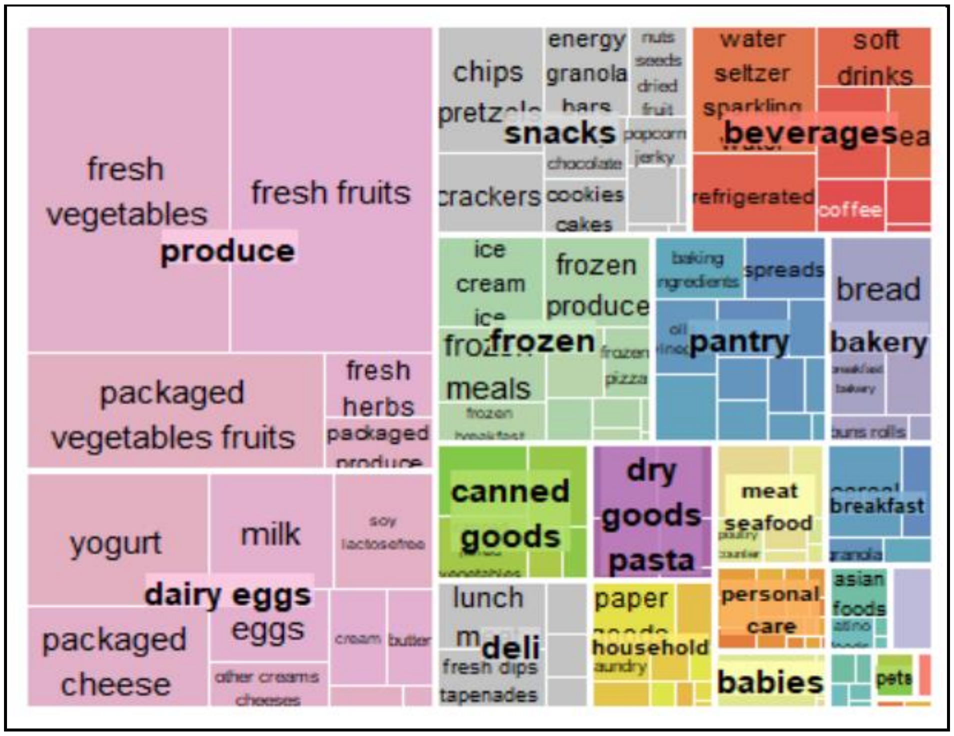
  Here I used the treemap package to visualize the structure of InstaCart product portfolio. In total there are 21 departments containing 134 aisles

- **How are aisles organized within departments?**



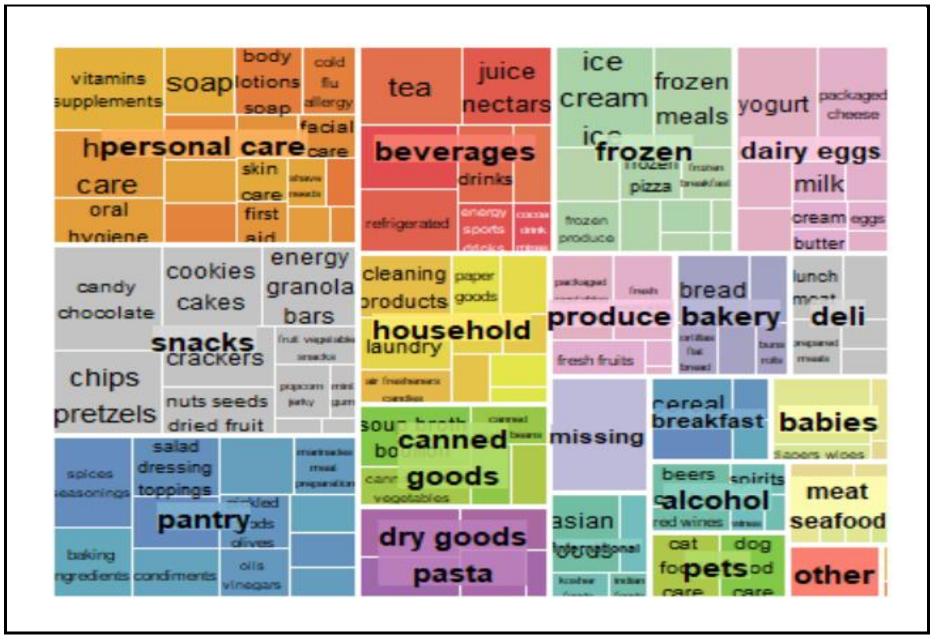
- **How many unique products are offered in each department/aisle?**

  The size of the boxes shows the number of products in each category.



- **How often are products from the department/aisle sold?**

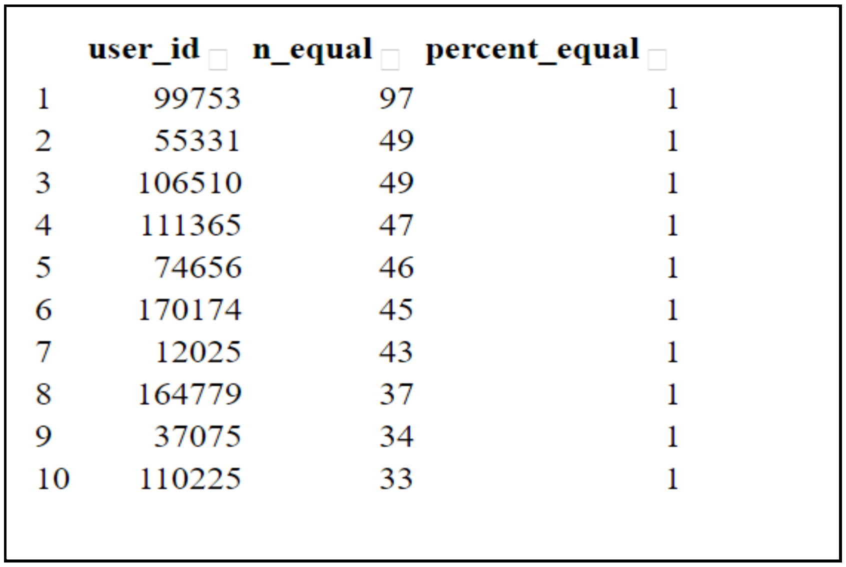
  The size of the boxes shows the number of sales.



- **Exploring Customer Habits**

  Here I can see for customers who just reorder the same products again all the time. To search these, I looked at all orders (excluding the first order), where the percentage of reordered items is exactly 1 (This can easily be adapted to look at more lenient thresholds). We can see there are in fact **3,487** customers, just always reordering products.

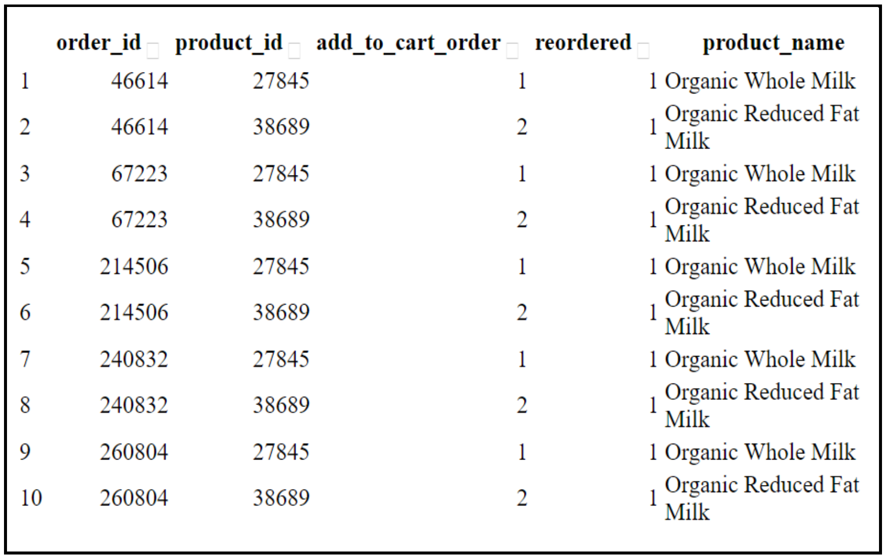
- **Customers reordering only:**



	user_id	n_equal	percent_equal
1	99753	97	1
2	55331	49	1
3	106510	49	1
4	111365	47	1
5	74656	46	1
6	170174	45	1
7	12025	43	1
8	164779	37	1
9	37075	34	1
10	110225	33	1

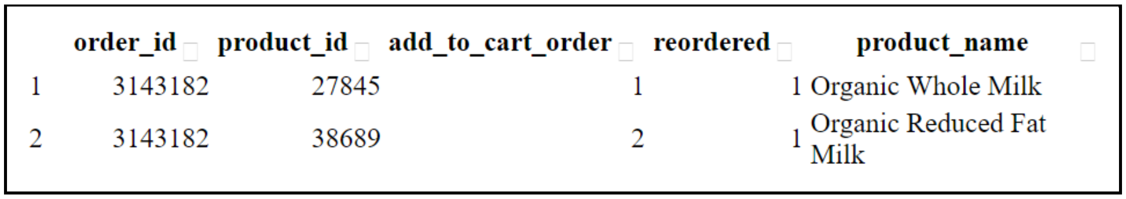
- **The customer with the strongest habit**

  The coolest customer is id #99753, having 97 orders with only reordered items. That is what I call a strong habit. She/he seems to like Organic Milk.



	order_id	product_id	add_to_cart_order	reordered	product_name
1	46614	27845	1	1	Organic Whole Milk
2	46614	38689	2	1	Organic Reduced Fat Milk
3	67223	27845	1	1	Organic Whole Milk
4	67223	38689	2	1	Organic Reduced Fat Milk
5	214506	27845	1	1	Organic Whole Milk
6	214506	38689	2	1	Organic Reduced Fat Milk
7	240832	27845	1	1	Organic Whole Milk
8	240832	38689	2	1	Organic Reduced Fat Milk
9	260804	27845	1	1	Organic Whole Milk
10	260804	38689	2	1	Organic Reduced Fat Milk

Let us look at his order in the train set. One would assume that he would buy “Organic Whole “Milk” and “Organic Reduced Fat Milk”:



	order_id	product_id	add_to_cart_order	reordered	product_name
1	3143182	27845	1	1	Organic Whole Milk
2	3143182	38689	2	1	Organic Reduced Fat Milk

5.Predictive Analysis

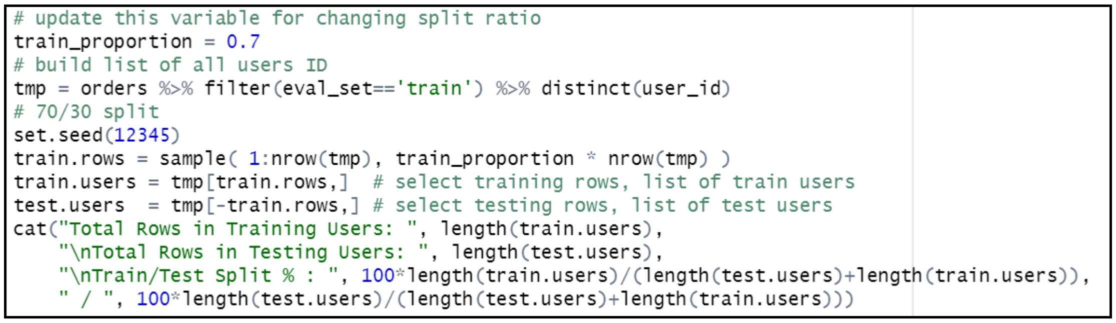
5.1 Type of Prediction

The objective is to predict what product will the customer purchase in the next basket. It requires probability estimation of each product that has been purchased before, that to be purchased before. **This is a classification problem**, as well as **a regression of probability** of repurchases.

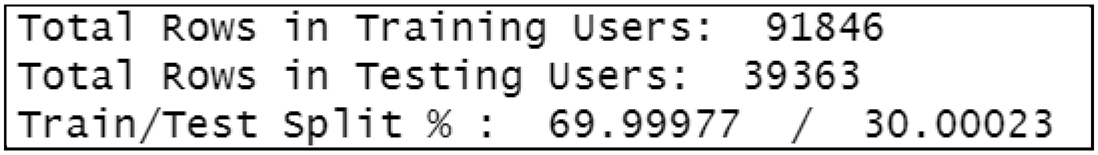
For this analysis, I have use two Naive models (handcrafted baseline) and one Machine Learning Logistic regression for Machine Learning approach for its speed and simplicity; to demonstrate the feasibility to producing a better outcome then baseline.

5.1.1 Train/Test Dataset Splitting

Instacart did not provide us **test order detail**, therefore we shall use the **train** users for both training and testing. We achieve this by splitting the **train** users and its related orders and products into train dataset and train dataset, at 70%/30% split (by number of users). That means our train/test dataset will contain approximately 91846 / 39,363 users.



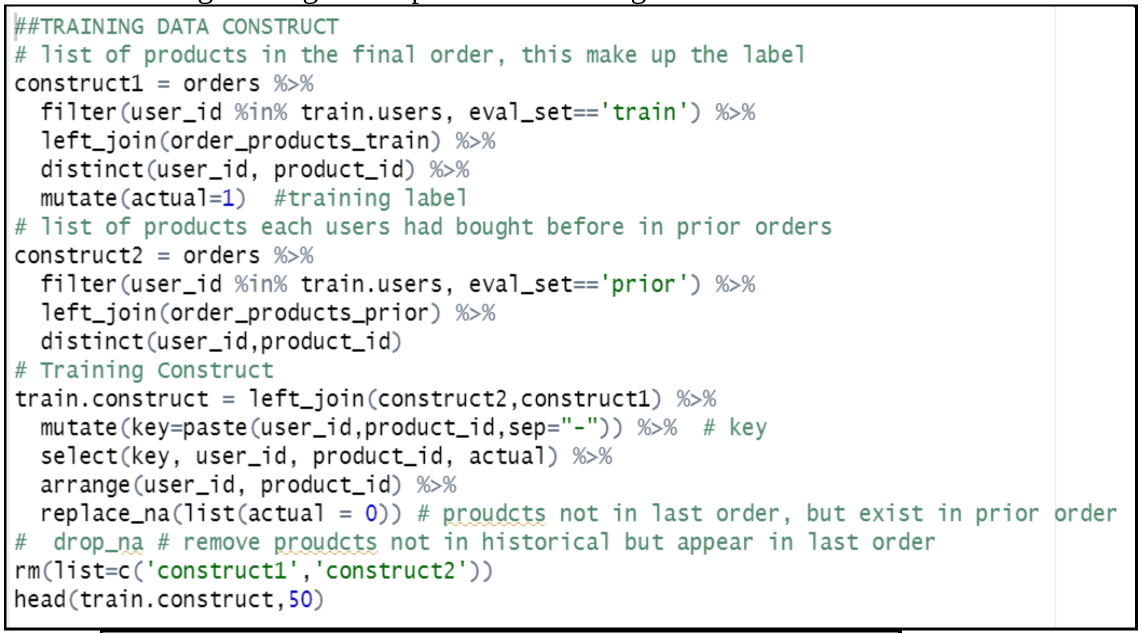
```
# update this variable for changing split ratio
train_proportion = 0.7
# build list of all users ID
tmp = orders %>% filter(eval_set=='train') %>% distinct(user_id)
# 70/30 split
set.seed(12345)
train.rows = sample( 1:nrow(tmp), train_proportion * nrow(tmp) )
train.users = tmp[train.rows,]  # select training rows, list of train users
test.users  = tmp[-train.rows,] # select testing rows, list of test users
cat("Total Rows in Training Users: ", length(train.users),
    "\nTotal Rows in Testing Users: ", length(test.users),
    "\nTrain/Test Split % : ", 100*length(train.users)/(length(test.users)+length(train.users)),
    " / ", 100*length(test.users)/(length(test.users)+length(train.users)))
```



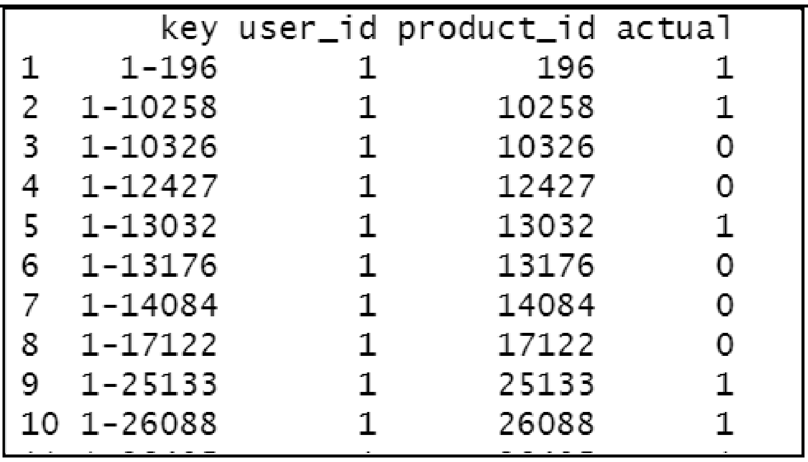
```
Total Rows in Training Users:  91846
Total Rows in Testing Users:  39363
Train/Test Split % :  69.99977  /  30.00023
```

5.1.2 Training Data Construct

The data frame used for training should contain the below columns and features:
Key: This is unique pair of user_id and product_id from orders
The keys should be constructed from all user_id-product_id pair that includes all prior and test/train rows
Actual: This is the response variable with value of 1 or 0 for each unique key
The value is 1 when the product is purchased in the last order (train or test set of orders)
The value is 0 when the product is not purchased in the train or test set, but was bought in prior set
other features: From exploratory discovery, features that could contribute to the prediction should be populated into the construct. Feature engineering will happen in the later stage.
Let's proceed to create the basic **training construct**. This won't be used for prediction until feature engineering is completed in later stage



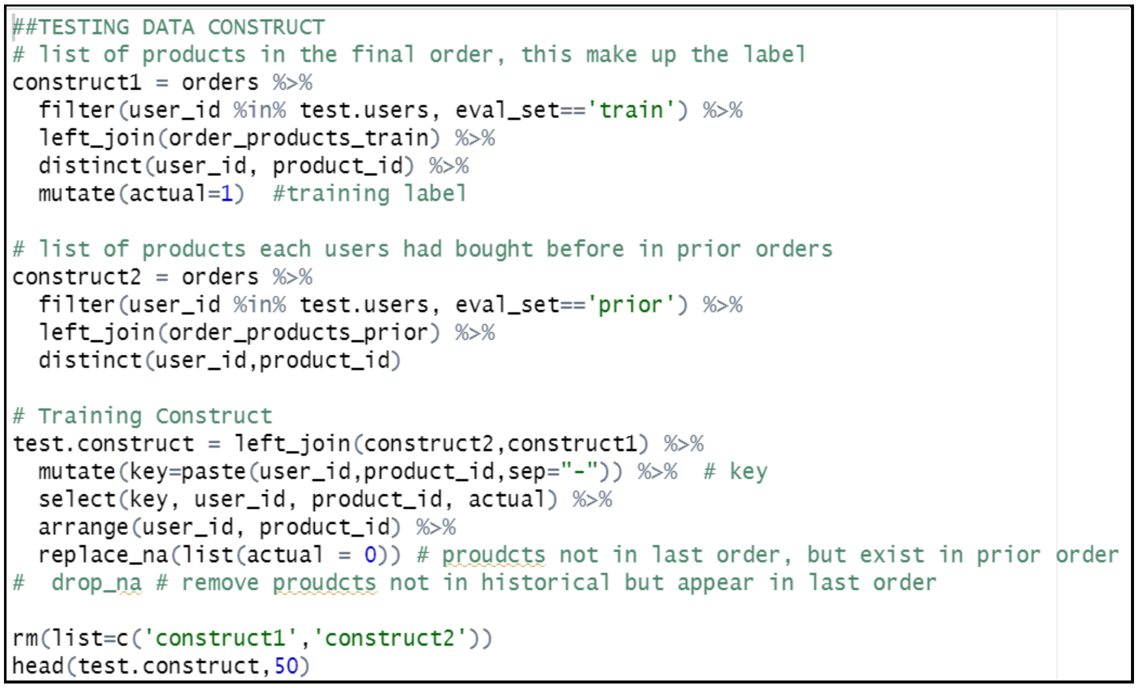
```
##TRAINING DATA CONSTRUCT
# list of products in the final order, this make up the label
construct1 = orders %>%
  filter(user_id %in% train.users, eval_set=='train') %>%
  left_join(order_products_train) %>%
  distinct(user_id, product_id) %>%
  mutate(actual=1)  #training label
# list of products each users had bought before in prior orders
construct2 = orders %>%
  filter(user_id %in% train.users, eval_set=='prior') %>%
  left_join(order_products_prior) %>%
  distinct(user_id,product_id)
# Training Construct
train.construct = left_join(construct2,construct1) %>%
  mutate(key=paste(user_id,product_id,sep="-")) %>%  # key
  select(key, user_id, product_id, actual) %>%
  arrange(user_id, product_id) %>%
  replace_na(list(actual = 0)) # proudcts not in last order, but exist in prior order
#  drop_na # remove proudcts not in historical but appear in last order
rm(list=c('construct1','construct2'))
head(train.construct,50)
```



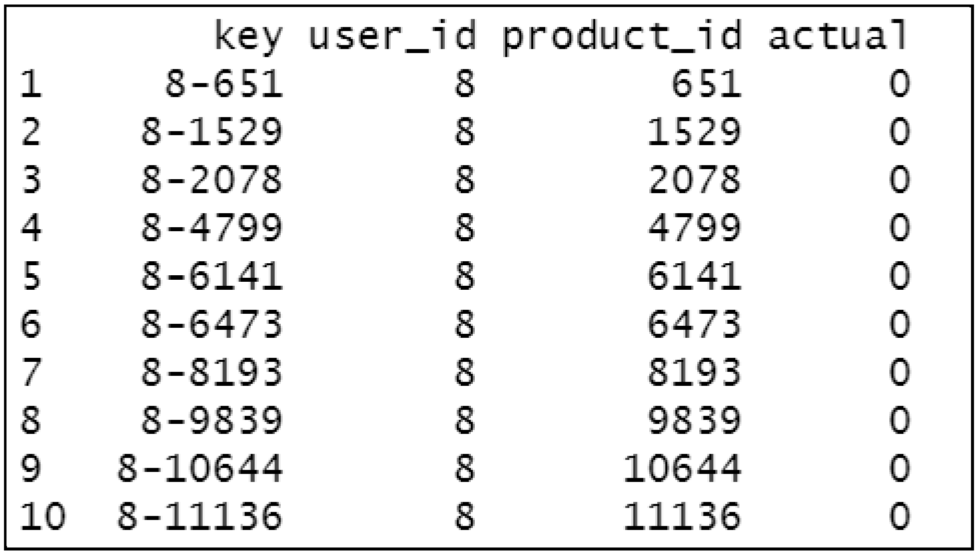
```
        key user_id product_id actual
1     1-196       1        196      1
2   1-10258       1      10258      1
3   1-10326       1      10326      0
4   1-12427       1      12427      0
5   1-13032       1      13032      1
6   1-13176       1      13176      0
7   1-14084       1      14084      0
8   1-17122       1      17122      0
9   1-25133       1      25133      1
10  1-26088       1      26088      1
```

5.1.3 Testing Data Construct

Similar approach to training data construct, here we frame the testing data for evaluate our model built with training data



```
##TESTING DATA CONSTRUCT
# list of products in the final order, this make up the label
construct1 = orders %>%
  filter(user_id %in% test.users, eval_set=='train') %>%
  left_join(order_products_train) %>%
  distinct(user_id, product_id) %>%
  mutate(actual=1)  #training label

# list of products each users had bought before in prior orders
construct2 = orders %>%
  filter(user_id %in% test.users, eval_set=='prior') %>%
  left_join(order_products_prior) %>%
  distinct(user_id,product_id)

# Training Construct
test.construct = left_join(construct2,construct1) %>%
  mutate(key=paste(user_id,product_id,sep="-")) %>%  # key
  select(key, user_id, product_id, actual) %>%
  arrange(user_id, product_id) %>%
  replace_na(list(actual = 0)) # proudcts not in last order, but exist in prior order
#  drop_na # remove proudcts not in historical but appear in last order

rm(list=c('construct1','construct2'))
head(test.construct,50)
```



```
          key user_id product_id actual
1       8-651       8        651      0
2      8-1529       8       1529      0
3      8-2078       8       2078      0
4      8-4799       8       4799      0
5      8-6141       8       6141      0
6      8-6473       8       6473      0
7      8-8193       8       8193      0
8      8-9839       8       9839      0
9     8-10644       8      10644      0
10    8-11136       8      11136      0
```

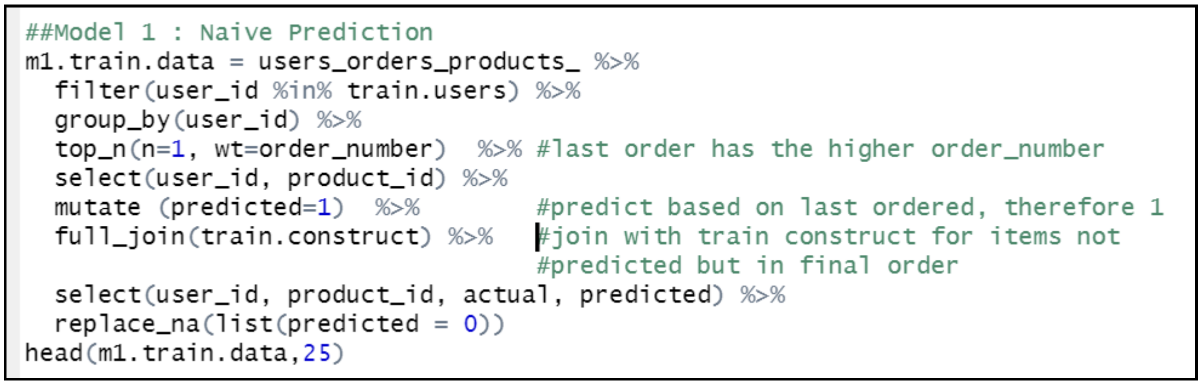
5.2 Model Evaluation & Optimization

Instacart has close to 50k products in their catalogue. As the maximum number of items ordered by a user is just a fraction of the 50k available product. This means by simply predicting nothing is purchased in the next basket, we would yield **close to 100% accuracy**.
Due to the **highly imbalance** dataset, Instacart require **F1 Score** as the competition scoring, instead of accuracy.

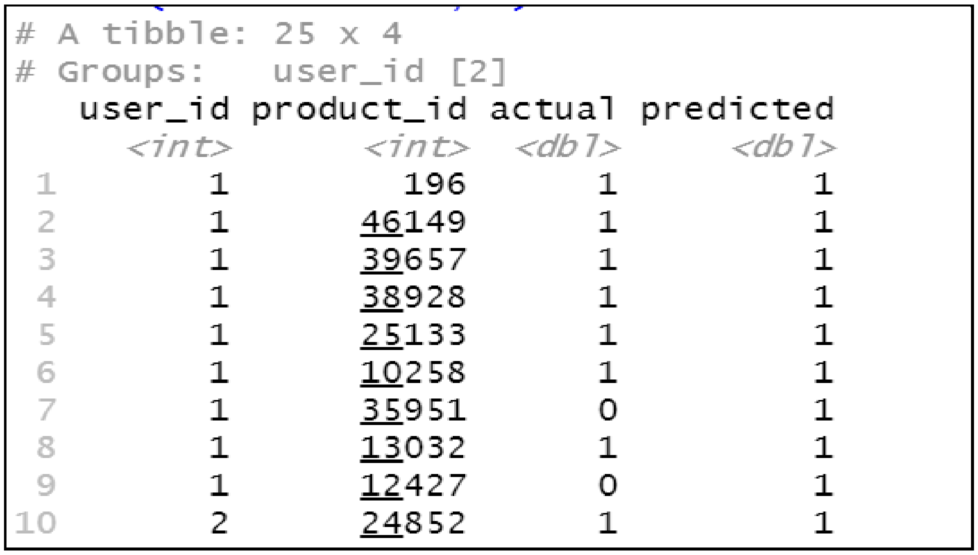
To evaluate the performance of the model, we had created a custom function to build a **confusion matrix and derive other binary classification metrics**.

5.3 Model 1: Naive Prediction

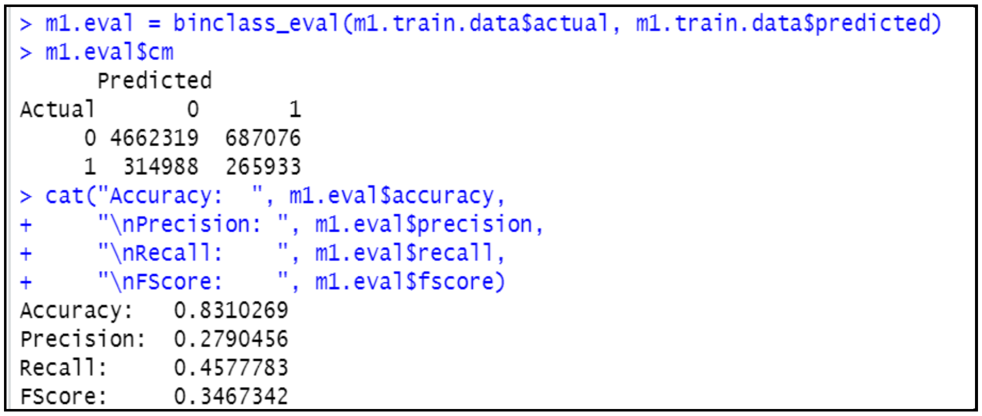
With intension to make this a baseline model, simply predict the basket based on user last order.



```
##Model 1 : Naive Prediction
m1.train.data = users_orders_products_ %>%
  filter(user_id %in% train.users) %>%
  group_by(user_id) %>%
  top_n(n=1, wt=order_number)  %>% #last order has the higher order_number
  select(user_id, product_id) %>%
  mutate (predicted=1)  %>%       #predict based on last ordered, therefore 1
  full_join(train.construct) %>%   #join with train construct for items not
                                   #predicted but in final order
  select(user_id, product_id, actual, predicted) %>%
  replace_na(list(predicted = 0))
head(m1.train.data,25)
```



```
# A tibble: 25 x 4
# Groups:   user_id [2]
   user_id product_id actual predicted
     <int>      <int>  <dbl>     <dbl>
 1       1        196      1         1
 2       1      46149      1         1
 3       1      39657      1         1
 4       1      38928      1         1
 5       1      25133      1         1
 6       1      10258      1         1
 7       1      35951      0         1
 8       1      13032      1         1
 9       1      12427      0         1
10       2      24852      1         1
```

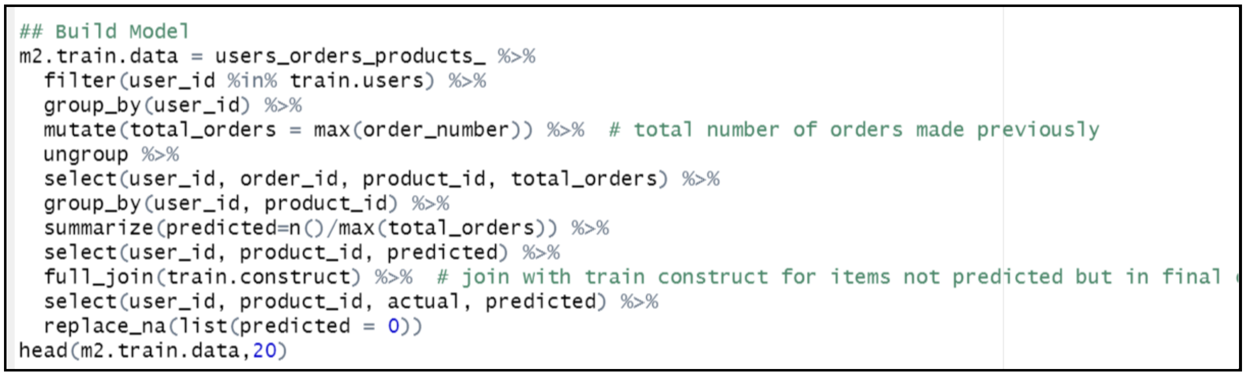


```
> m1.eval = binclass_eval(m1.train.data$actual, m1.train.data$predicted)
> m1.eval$cm
      Predicted
Actual       0       1
     0 4662319  687076
     1  314988  265933
> cat("Accuracy:  ", m1.eval$accuracy,
+     "\nPrecision: ", m1.eval$precision,
+     "\nRecall:    ", m1.eval$recall,
+     "\nFScore:    ", m1.eval$fscore)
Accuracy:   0.8310269
Precision:  0.2790456
Recall:     0.4577783
FScore:     0.3467342
```

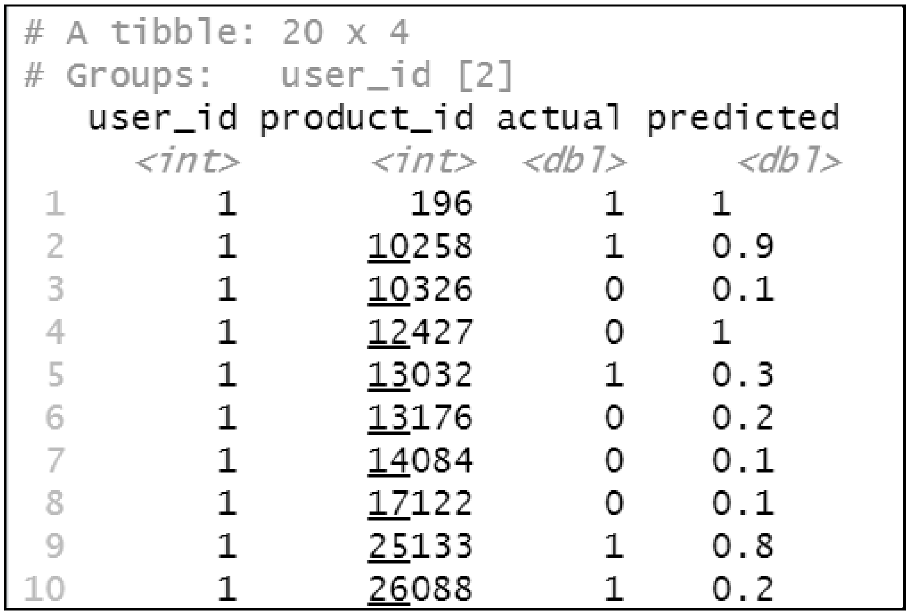
The result shows only **0.3460833 F1 Score**.

5.4 Model 2: Smarter Naive Prediction (Baseline)

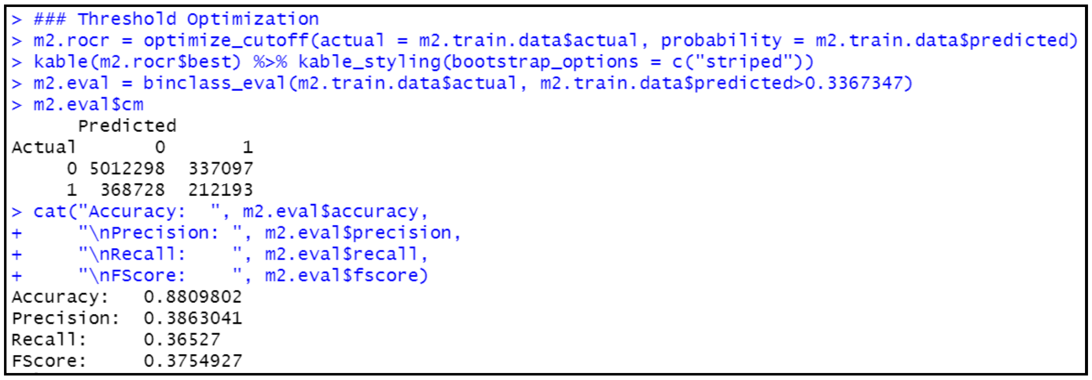
In this model, we predict products in the basket by estimating their frequency of repurchased. This way we get a ratio to indicate probability of re-purchases. We use ROCR package to estimate the best cutoff point (at which above this cutoff we shall predict for re-order) that give us the **optimum F1 score**.



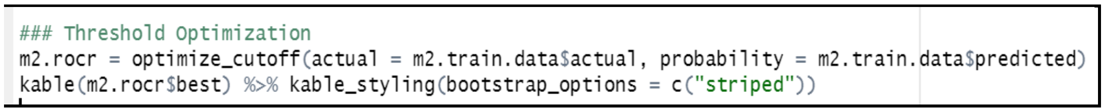
```
## Build Model
m2.train.data = users_orders_products_ %>%
  filter(user_id %in% train.users) %>%
  group_by(user_id) %>%
  mutate(total_orders = max(order_number)) %>%  # total number of orders made previously
  ungroup %>%
  select(user_id, order_id, product_id, total_orders) %>%
  group_by(user_id, product_id) %>%
  summarize(predicted=n()/max(total_orders)) %>%
  select(user_id, product_id, predicted) %>%
  full_join(train.construct) %>%  # join with train construct for items not predicted but in final
  select(user_id, product_id, actual, predicted) %>%
  replace_na(list(predicted = 0))
head(m2.train.data,20)
```



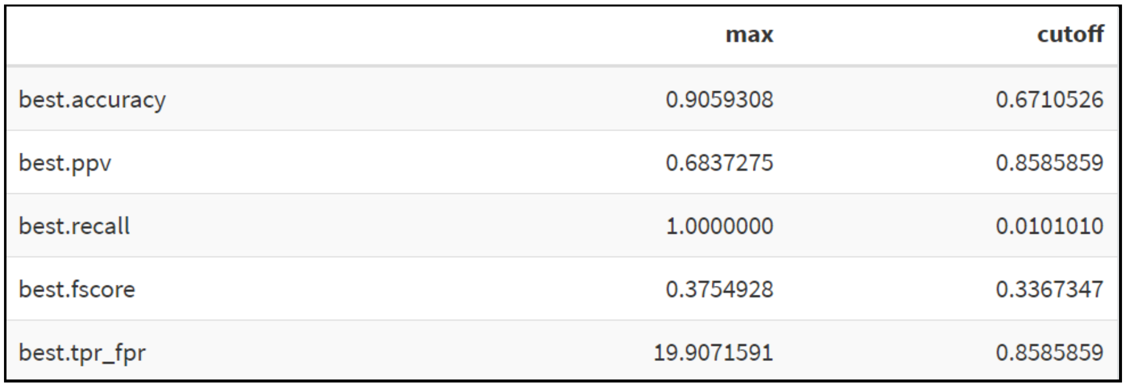
```
# A tibble: 20 x 4
# Groups:   user_id [2]
   user_id product_id actual predicted
     <int>      <int>  <dbl>     <dbl>
 1       1        196      1     1
 2       1      10258      1     0.9
 3       1      10326      0     0.1
 4       1      12427      0     1
 5       1      13032      1     0.3
 6       1      13176      0     0.2
 7       1      14084      0     0.1
 8       1      17122      0     0.1
 9       1      25133      1     0.8
10       1      26088      1     0.2
```



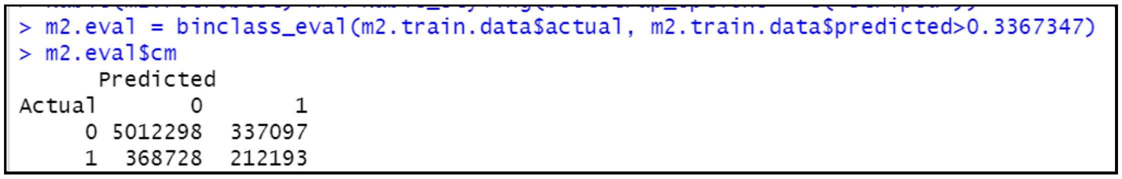
```
> ### Threshold Optimization
> m2.rocr = optimize_cutoff(actual = m2.train.data$actual, probability = m2.train.data$predicted)
> kable(m2.rocr$best) %>% kable_styling(bootstrap_options = c("striped"))
> m2.eval = binclass_eval(m2.train.data$actual, m2.train.data$predicted>0.3367347)
> m2.eval$cm
      Predicted
Actual       0       1
     0 5012298  337097
     1  368728  212193
> cat("Accuracy:  ", m2.eval$accuracy,
+     "\nPrecision: ", m2.eval$precision,
+     "\nRecall:    ", m2.eval$recall,
+     "\nFScore:    ", m2.eval$fscore)
Accuracy:   0.8809802
Precision:  0.3863041
Recall:     0.36527
FScore:     0.3754927
```



```
### Threshold Optimization
m2.rocr = optimize_cutoff(actual = m2.train.data$actual, probability = m2.train.data$predicted)
kable(m2.rocr$best) %>% kable_styling(bootstrap_options = c("striped"))
```

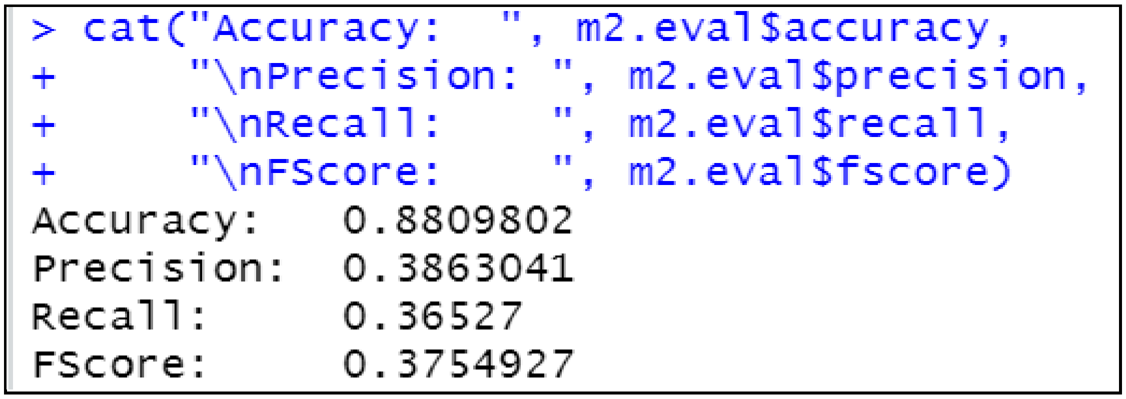


	max	cutoff
best.accuracy	0.9059308	0.6710526
best.ppv	0.6837275	0.8585859
best.recall	1.0000000	0.0101010
best.fscore	0.3754928	0.3367347
best.tpr_fpr	19.9071591	0.8585859



```
> m2.eval = binclass_eval(m2.train.data$actual, m2.train.data$predicted>0.3367347)
> m2.eval$cm
      Predicted
Actual       0       1
     0 5012298  337097
     1  368728  212193
```

We are getting slightly **better F1 Score (0.3753544)** compare to previous naive model. We shall use this as the **BASELINE**.



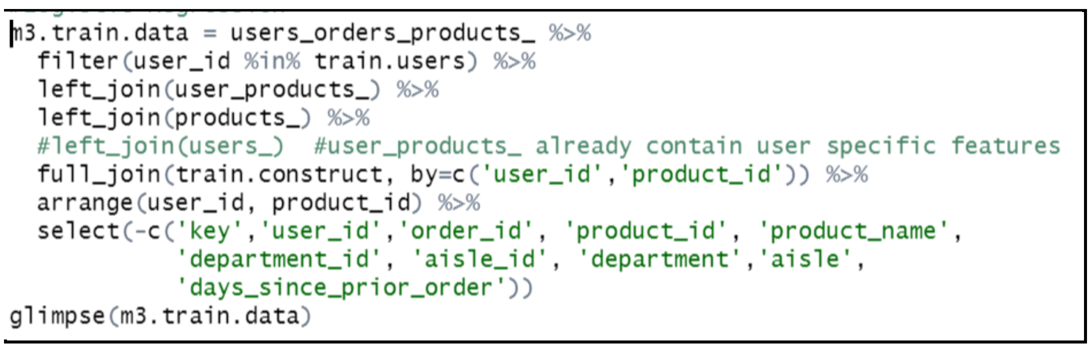
```
> cat("Accuracy:  ", m2.eval$accuracy,
+     "\nPrecision: ", m2.eval$precision,
+     "\nRecall:    ", m2.eval$recall,
+     "\nFScore:    ", m2.eval$fscore)
Accuracy:   0.8809802
Precision:  0.3863041
Recall:     0.36527
FScore:     0.3754927
```

5.5 Machine Learning Framing

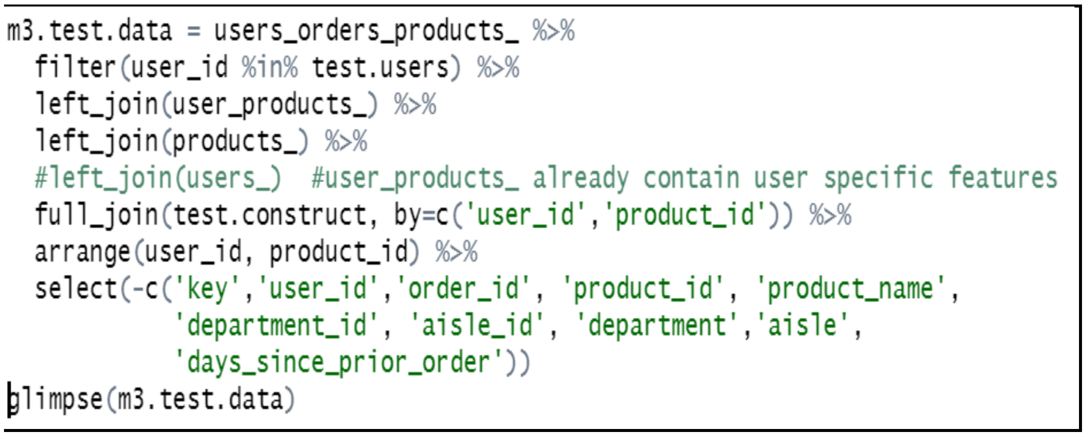
We construct all the products that users had purchased in the last 3 orders, then use machine learning classification to predict will each of the product be purchased again. We shall use **decision tree and logistic regression** for this prediction.

5.6 Model 3: Logistic Regression

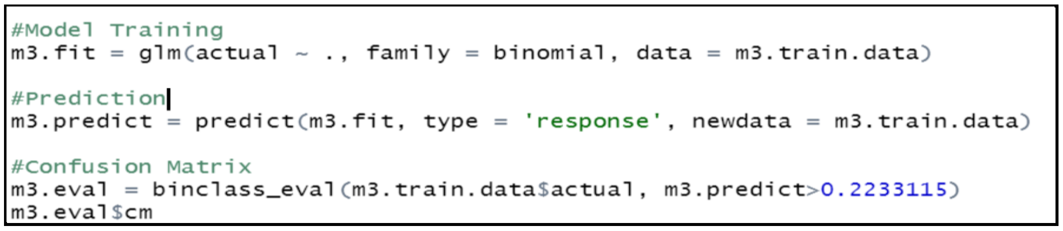
Constructing Training and Testing Data



```
m3.train.data = users_orders_products_ %>%
  filter(user_id %in% train.users) %>%
  left_join(user_products_) %>%
  left_join(products_) %>%
  #left_join(users_)  #user_products_ already contain user specific features
  full_join(train.construct, by=c('user_id','product_id')) %>%
  arrange(user_id, product_id) %>%
  select(-c('key','user_id','order_id', 'product_id', 'product_name',
            'department_id', 'aisle_id', 'department','aisle',
            'days_since_prior_order'))
glimpse(m3.train.data)
```

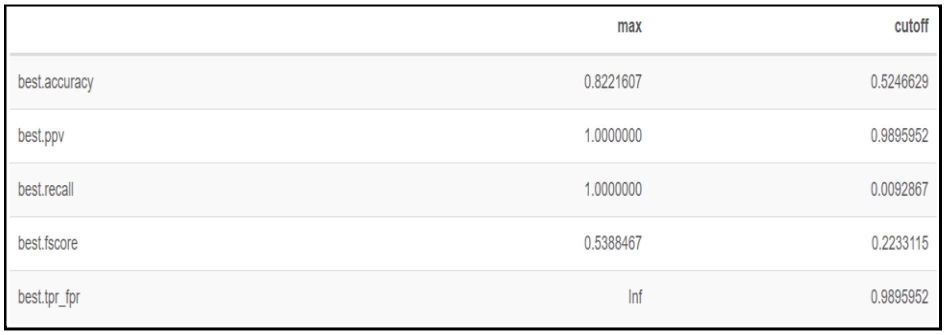


```
m3.test.data = users_orders_products_ %>%
  filter(user_id %in% test.users) %>%
  left_join(user_products_) %>%
  left_join(products_) %>%
  #left_join(users_)  #user_products_ already contain user specific features
  full_join(test.construct, by=c('user_id','product_id')) %>%
  arrange(user_id, product_id) %>%
  select(-c('key','user_id','order_id', 'product_id', 'product_name',
            'department_id', 'aisle_id', 'department','aisle',
            'days_since_prior_order'))
glimpse(m3.test.data)
```



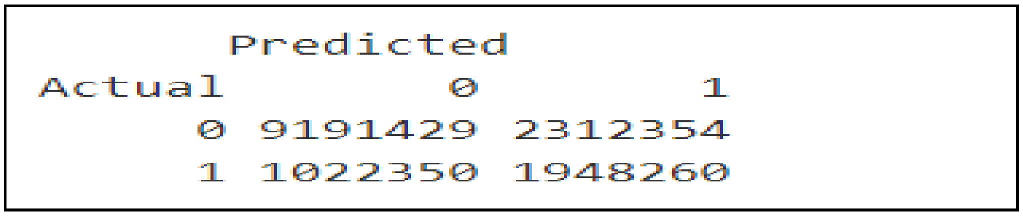
```
#Model Training
m3.fit = glm(actual ~ ., family = binomial, data = m3.train.data)

#Prediction
m3.predict = predict(m3.fit, type = 'response', newdata = m3.train.data)

#Confusion Matrix
m3.eval = binclass_eval(m3.train.data$actual, m3.predict>0.2233115)
m3.eval$cm
```

- **Training Data Performance**

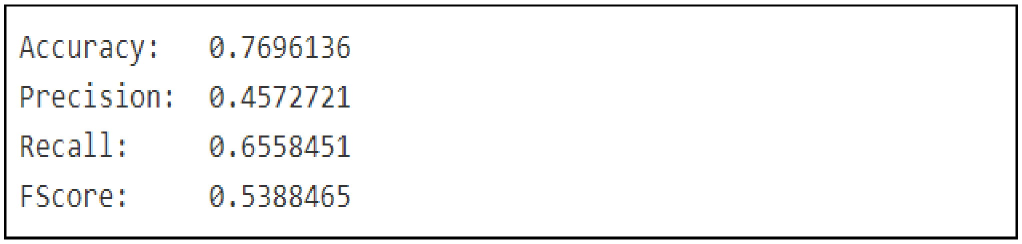


	max	cutoff
best.accuracy	0.8221607	0.5246629
best.ppv	1.0000000	0.9895952
best.recall	1.0000000	0.0092867
best.fscore	0.5388467	0.2233115
best.tpr_fpr	Inf	0.9895952

- **Model Evaluation**

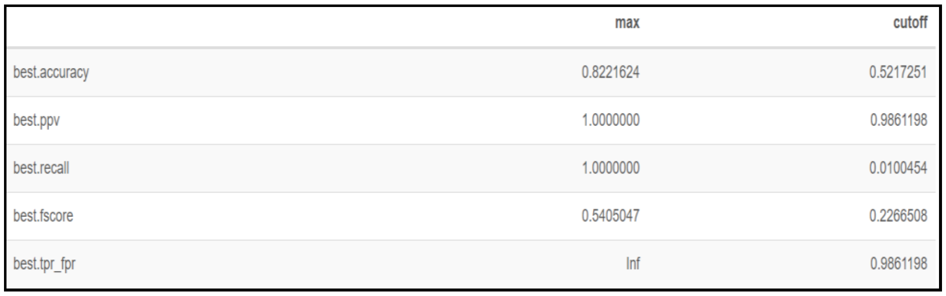


```
         Predicted
Actual          0        1
      0 9191429 2312354
      1 1022350 1948260
```

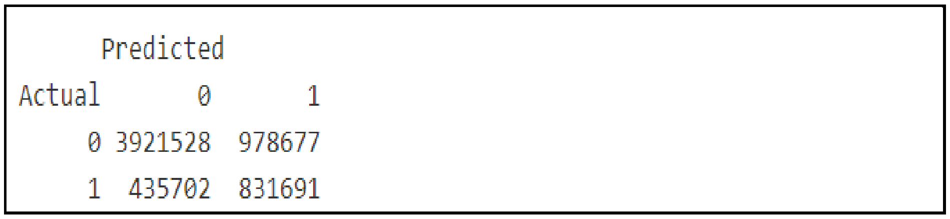


```
Accuracy:   0.7696136
Precision:  0.4572721
Recall:     0.6558451
FScore:     0.5388465
```

- **Test Data Performance**



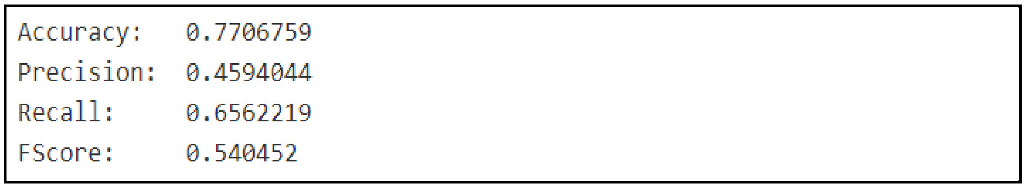
	max	cutoff
best.accuracy	0.8221624	0.5217251
best.ppv	1.0000000	0.9861198
best.recall	1.0000000	0.0100454
best.fscore	0.5405047	0.2266508
best.tpr_fpr	Inf	0.9861198



```
      Predicted
Actual       0      1
     0 3921528 978677
     1  435702 831691
```

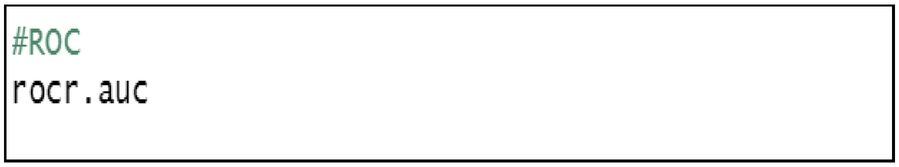
- **Model Evaluation**

  Logistic regression produces F1 Score of 0.5388937 with training data, a much better compared to Model 1 and Model 2. We shall proceed to test the model on unknown data, the test data.
  We achieved F1 Score of **0.5405588**, slightly higher than training data.

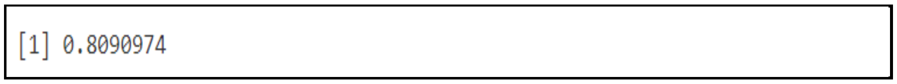


```
Accuracy:   0.7706759
Precision:  0.4594044
Recall:     0.6562219
FScore:     0.540452
```

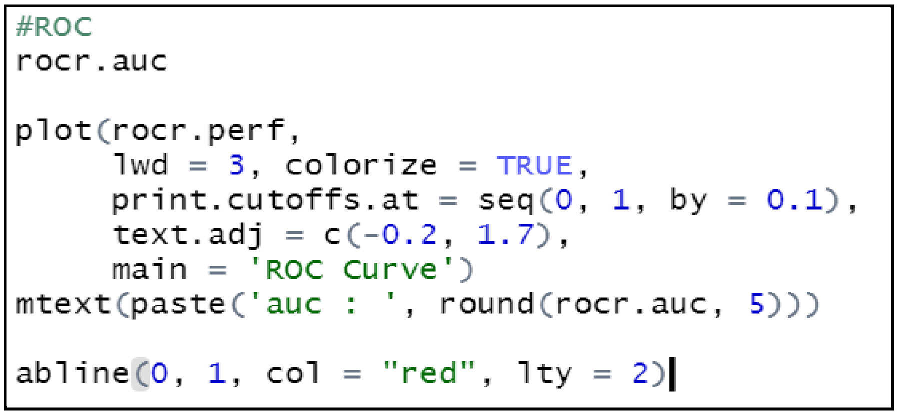
- **ROC**



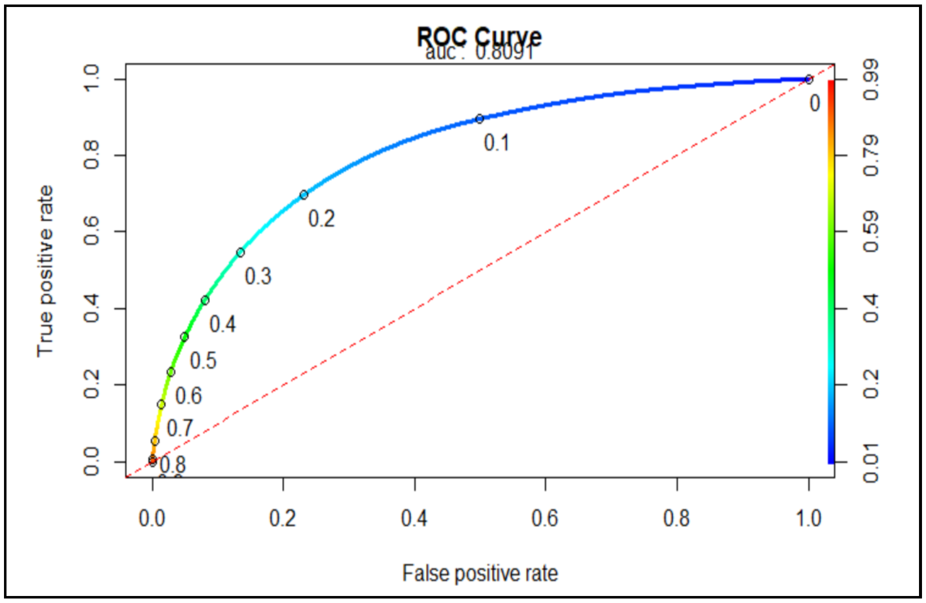
```
#ROC
rocr.auc
```



```
[1] 0.8090974
```



```
#ROC
rocr.auc

plot(rocr.perf,
     lwd = 3, colorize = TRUE,
     print.cutoffs.at = seq(0, 1, by = 0.1),
     text.adj = c(-0.2, 1.7),
     main = 'ROC Curve')
mtext(paste('auc : ', round(rocr.auc, 5)))

abline(0, 1, col = "red", lty = 2)
```



6.Conclusion

- Doing this analysis allowed me to understand the fine-grained details about the customer shopping behavior on the Instacart platform.
- Knowing which items are most frequently purchased is the first step for Instacart to optimize its software product and recommend items for customers while they shop.

7.Citations

- https://www.kaggle.com/c/instacart-market-basket-analysis/data
- http://blog.kaggle.com/2017/09/21/instacart-market-basket-analysis-winners-interview-2nd-place-kazuki-onodera/
- https://meiyipan.com/2017/09/16/instacart/
- http://cs229.stanford.edu/proj2017/final-reports/5240337.pdf